Digital Color Image Compression

With Real and Complex Artificial Neural Networks

Nick Handrick and Dr. Diana Thomson | Department of Mathematics

The Power of AND | University of Wisconsin Eau Claire

Abstract

Neural networks are an exciting and evolving branch of machine learning, but they are not limited to just artificial intelligence. Recently, they have been used to compress and even add digital watermarks, or copyright signatures, to digital images. Typically, neural networks use real numbers for their computations, but researchers have also experimented with using complex numbers and quaternions as the basis of these networks. Our research investigates the use of quaternion-valued neural networks implemented in Java for the purposes of digital image compression and watermarking. The benefits of using quaternion-valued over real-valued neural networks include faster network training time, better color compression/recovery, and less processing power/memory required for the computation.

Introduction

The process of compressing a digital image with a neural network can be explained in just a few steps. First, the image is parsed into a vector containing the color information for each pixel (figure 1). Second, a neural network is constructed with dimensions that match the image (figure 2). Third, the neural network is trained on the image until it can compress and reproduce the original image within a certain quality threshold. In practice, a significant amount of memory and computation are needed to compress large images, so this whole process is usually partitioned. That is, the image is broken into smaller pieces which can be individually compressed with their own smaller networks and then stitched back together (figure 3). Another benefit of partitioning is that the individual partitions can be trained in parallel, making use of many processors to speed up compression time. Watermarking the images involves a separate step which combines the compressed output with the "hidden layer" of the network.

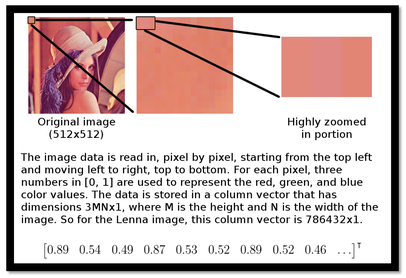

Figure 1: The relationship between images and input vectors used in the neural network.

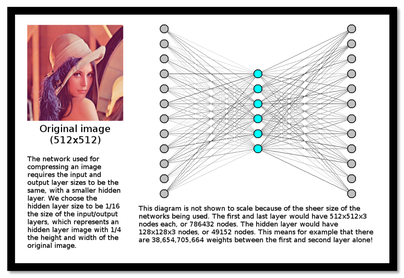

Figure 2: The dimensions of the neural network used in image compression/watermarking.

Figure 3: Partitioning the image data for more computationally feasible network training

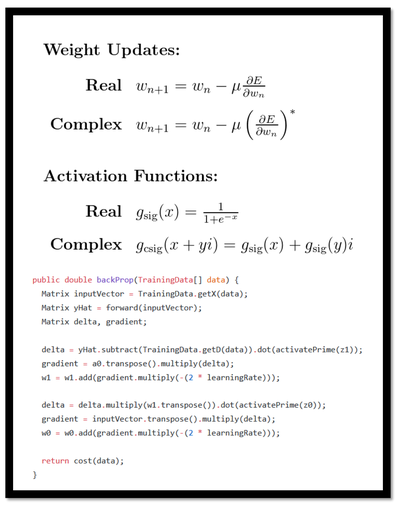

Figure 4: Various equations and code used in the neural network

Methods and Development Process

Watermarking with a quaternion valued neural network is a difficult task to start with. Instead, we made simpler networks first, and then iterated towards the desired end result. First, a real-valued neural network and matrix was implemented. The network was then tested on numerical data to make sure it was working properly. Next, code was written that can handle parsing and recovering images to/from vectors. This new "image utilities" code was used in conjunction with the neural network to successfully achieve digital image compression. After the real case was sorted out, a complex-valued neural network and corresponding complex matrix/image utilities were implemented. The complex backpropagation algorithm differed slightly from the real one. Also, only the real component was used for the image utilities. That is, the complex component was ignored in parsing/recovering pixel colors. Once the complex network succeeded in image compression, we began work on the quaternion network and corresponding matrix/image utilities. Around this time, we also started experimenting with using different activation functions. The idea was that different activation functions may be computationally more efficient, or they might train better.

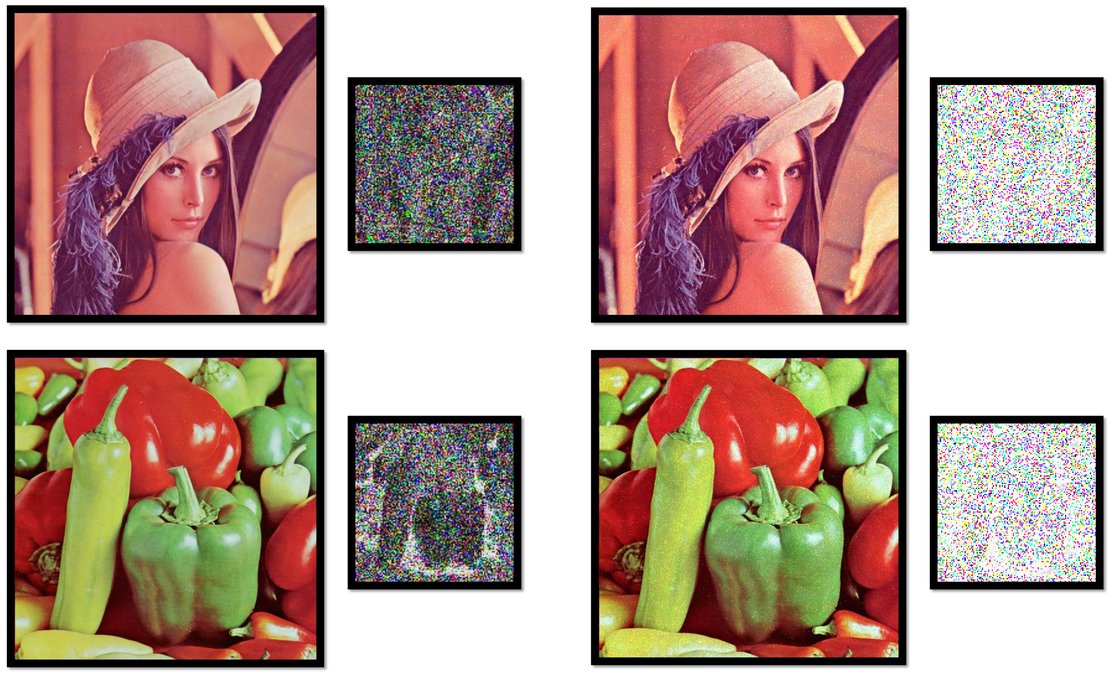

Figure 5: Some examples of images compressed with our networks. On the left are two 512x512 pixel images compressed with the real network (1000 iterations, 8x8 partitions), and on the right are the same two images compressed with the complex network (1000 iterations, 8x8 partitions). Notice how the complex images have more artifacts and worse compression quality for roughly the same amount of computation. Next to each image is the "hidden layer" (128x128 pixels, not to scale) recovered from the network after training. The hidden layer would be combined with the output in watermarking.

Future Work

A significant amount of time and effort were spent writing and debugging the real and complex networks. Since the quaternion network wasn't completed, that would be the next logical step to work on and finish. We didn't fully implement image watermarking, spending more time on getting the compression and networks working first. Also, we started experimenting with using different activation functions, including split functions and Taylor polynomial approximations.

Quaternions are a 4-dimensional kind of number with 3 imaginary parts and one real part. These are especially useful for image compression in a neural network because they give rise to a natural way to store color data. Rather than storing the RGB color values separately in 3 different, seemingly arbitrary numbers, they can each be stored in imaginary components of the same quaternion. For neural networks, this means that you can make a network with 1/3 as many nodes in the input/outputs layers, so 1/9 as many weights to keep track of. In theory, this should improve the network's ability to recover colors during compression/decompression.

Conclusion

Throughout working on this project, we effectively created code that can accomplish digital image compression for real numbers and complex numbers. The actual algorithms didn't change much between the real and complex networks, other than a few conjugates here and there (figure 4).

Also, the activation function needed to be modified to work with complex numbers. We decided on a split sigmoid function, but the code is written to be as general as possible, so any complex function with a known derivative can be easily implemented as an activation function.

As far as output, the complex network didn't perform as well as the real network. This makes sense because of how the imaginary component of the complex numbers is basically ignored, resulting in unused computation. All of that causes a slower network the accomplishes less in the same amount of time, and takes longer to train. Quaternions are expected to be more efficient though.

References

T. Isokawa, T. Kusakabe, N. Matsui, and F. Peper, "Quaternion neural network and its application," in Knowledge-Based Intelligent Information and Engineering Systems (V. Palade, R. J. Howlett, and L. Jain, eds.), vol. 2774 of Lecture Notes in Computer Science, pp. 318–324, Springer-Verlag Berlin Heidelberg, KES 2003.

T. Isokawa; H. Nishimura; N. Matsui, "Quaternionic Multilayer Perceptron with Local Analyticity." *Information* 2012, *3*, 756-770.

L. Luo, H. Feng, and L. Ding, "Color image compression based on quaternion neural network principal component analysis," in 2010 International Conference on Multimedia Technology (ICMT), IEEE, 2010.

H. Tarik and J. E. Miloud, "Digital watermarking and signing by artificial neural networks," *American Journal of Intelligent Systems*, vol. 4, no. 2, pp. 21–31, 2014.

D. Thomson La Corte, "Newton's Method Backpropagation for Complex-Valued Holomorphic Neural Networks: Algebraic and Analytic Properties" (2014). *Theses and Dissertations*. 565. https://dc.uwm.edu/etd/565

University of Southern California Signal and Image Processing Institute, "USC-SIPI image database." http://sipi.usc.edu/database/, September 2017.

We thank the Office of Research and Sponsored Programs for supporting this research, and Learning & Technology Services for printing this poster.