Quaternion Convolutional Neural Networks

Xuanyu Zhu   Yi Xu   Hongteng Xu   Changjian Chen

January 25, 2022

Table of Content

Introduction

As a powerful feature representation method, convolutional neural networks (CNNs) have been widely applied in the field of computer vision.

One key module of CNN model is the convolution layer, which extracts features from high-dimensional structural data efficiently by a set of convolution kernels.

When dealing with multi-channel inputs (e.g., color images), the convolution kernels merges these channels by summing up the convolution results and output one single channel per kernel accordingly.

Introduction

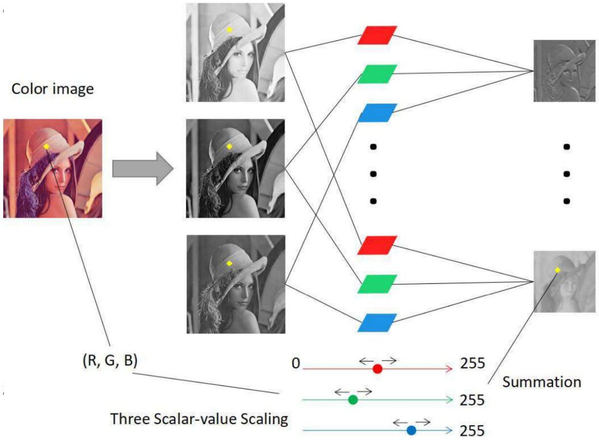

Figure: Real-valued CNN

Introduction

We may lose important structural information of color and obtain non-optimal representation of color image.

Simply summing up the outputs gives too many degrees of freedom to the learning of convolution kernels.

We may have a high risk of over-fitting even if imposing heavy regularization terms.

Introduction

We propose a novel quaternion convolutional neural network (QCNN) model, which represents color image in the quaternion domain.

While the traditional real-valued convolution is only capable to enforce scaling transformation on the input, specifically, the quaternion convolution achieves the scaling and the rotation of input in the color space, which provides us with more structural representation of color information.

We establish fully-quaternion CNNs to represent color images in a more effective way and study the relationship between our QCNN model and existing real-valued CNNs and find a compatible way to combine them together in a same algorithmic framework.

Introduction

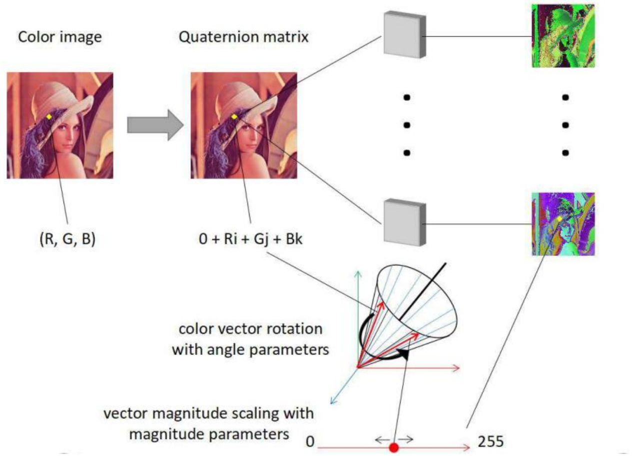
Color image
Quaternion matrix
(R, G, B)
0 + Ri + Gj + Bk
color vector rotation with angle parameters
vector magnitude scaling with magnitude parameters
0
255

Figure: Quaternion CNN

Related works

Quaternion-based color image processing

A quaternion $\hat{q}$ in the quaternion domain $\mathbb{H}$, *i.e*, $q \in \mathbb{H}$, can be represented as $\hat{q} = q_0 + q_1 i + q_2 j + q_3 k$, where $q_i \in \mathbb{R}$ for $i = 0, 1, 2, 3$, and the imaginary units i, j, k obey the quaternion rules that $i^2 = j^2 = k^2 = ijk = -1$. Similar to real numbers, we can define a series of operations for quaternions:

Related works

Quaternion-based color image processing

- Addition:

$$\begin{aligned}\hat{p}+\hat{q}=&(p_0+q_0)+(p_1+q_1)i\\&+(p_2+q_2)j+(p_3+q_3)k\end{aligned}$$

- Scalar multiplication:

$$\lambda\hat{q}=\lambda q_0+\lambda q_1 i+\lambda q_2 j+\lambda q_3 k$$

- Element multiplication:

$$\begin{aligned}\hat{p}\hat{q}=&(p_0q_0-p_1q_1-p_2q_2-p_3q_3)\\&+(p_0q_1+p_1q_0+p_2q_3-p_3q_2)i\\&+(p_0q_2-p_1q_3-p_2q_0-p_3q_1)j\\&+(p_0q_3+p_1q_2-p_2q_1-p_3q_0)k\end{aligned}$$

- Conjugation:

$$\hat{q}^* = q_0 - q_1 i - q_2 j - q_3 k$$

These quaternion operations can be used to represent rotations in a three-dimensional space. Suppose that we rotate a 3D vector $\mathbf{q} = [q_1, q_2, q_3]^T$ to get new vector $\mathbf{p} = [p_1, p_2, p_3]^T$, with an angle θ and along a rotation axis $\mathbf{w} = [w_1, w_2, w_3]^T$, $w_1^2 + w_2^2 + w_3^2 = 1$. Such a rotation is equivalent to the following quaternion operation:

$$\hat{p} = \hat{w}\hat{q}\hat{w}^* \quad (1)$$

where $\hat{q} = 0 + q_1 i + q_2 j + q_3 k$, $\hat{p} = 0 + p_1 i + p_2 j + p_3 k$ and

$$\hat{w} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}(w_1 i + w_2 j + w_3 k) \quad (2)$$

Related works

Real-valued CNNs and their extensions

Convolutional neural network is one of the most successful models in many vision tasks. CNNs have achieved encouraging results in many field like digit recognition, image classification, low-level vision.

Some efforts have been made to extend real-valued neural networks to other number fields. Complex-valued neural networks have been built and proved to have advantage on generalization ability and can be more easily optimized.

In *Deep quaternion networks*, a deep quaternion network is proposed. However, its convolution simply replaces the real multiplications with quaternion ones, and its quaternion kernel is not further parameterized. Our proposed quaternion convolution, however, is physically-meaningful for color image processing tasks.

Proposed Quaternion CNNs

Quaternion convolution layers

Focusing on color image representation, our quaternion CNN treats a color image as a 2D pure quaternion matrix, denoted as $\hat{A} = [\hat{a}_{nn'}] \in \mathbb{H}^{N\times N}$ where N represents the size of the image. In particular, the quaternion matrix $\hat{A}$ is

$$\hat{A} = 0 + \mathbf{R}i + \mathbf{G}j + \mathbf{B}k, \tag{3}$$

where $\mathbf{R}, \mathbf{G}, \mathbf{B} \in \mathbb{R}^{N\times N}$ represent red, green and blue channels, respectively.

Proposed Quaternion CNNs

Quaternion convolution layers

Suppose that we have an $L \times L$ quaternion convolution kernel $\hat{W} = [\hat{w}_{ll'}] \in \mathbb{H}^{L\times L}$. We aim to design an effective and physically-meaningful quaternion convolution operation, denoted as $\circledast$ between the input $\hat{A}$ and kernel $\hat{W}$.

Specifically, this operation should

1. apply rotations and scalings to color vectors in order to find the best representation in the whole color space;
2. play the same role as real-valued convolution when processing grayscale images.

Proposed Quaternion CNNs

Quaternion convolution layers

To achieve this aim, we take advantage of the rotational nature of quaternion shown in (1,2) and propose a quaternion convolution in a particular form. Specifically, we set the element of the quaternion convolution kernel as

$$\hat{w}_{ll'} = s_{ll'}(\cos\frac{\theta_{ll'}}{2} + \sin\frac{\theta_{ll'}}{2}\mu) \tag{4}$$

where $\theta_{ll'} \in [-\pi, \pi]$ and $s_{ll'} \in \mathbb{R}$. μ is the gray axis with unit length *i.e*, $\frac{\sqrt{3}}{3}(i + j + k)$.

Proposed Quaternion CNNs

Quaternion convolution layers

The quaternion convolution is defined as

$$\hat{A} \circledast \hat{W} = \hat{F} = [\hat{f}_{kk'}] \in \mathbb{H}^{(N-L+1)\times(N-L+1)} \tag{5}$$

where

$$\hat{f}_{kk'} = \sum_{l=1}^{L}\sum_{l'=1}^{L} \frac{1}{s_{ll'}} \hat{w}_{ll'} \hat{a}_{(k+l)(k'+l')} \hat{w}_{ll'}^{*} \tag{6}$$

For our QCNNs, one pixel is a quaternion or a 3D vector in color space, the proposed convolution find its best representation in a small part of the color space because we restrict the convolution to apply only a rotate and a scaling transform.

Such a convolution actually impose implicit regularizers on the model, such that we can suppress the risk of over-fitting brought by too many degrees of freedom to the learning of kernels.

Proposed Quaternion CNNs

Quaternion convolution layers

Although our convolution operation is designed for color image, it can be applied to grayscale image as well. For grayscale images, they can be seen as color images whose channels are the same. Because all the corresponding color vectors are parallel to the gray axis, the rotate transform equals to identical transformation, thus the quaternion convolution performs the same function as real-valued convolution. From this viewpoint, real-valued convolution is a special case of quaternion convolution for grayscale image.

Proposed Quaternion CNNs
Quaternion convolution layers

According to the rule of quaternion computations, if we represent each $\hat{a}_{nn'}$, as a 3D vector $\mathbf{a}_{nn'} = [a_1, a_2, a_3]^T$, then the operation in (6) can be represented by a set of matrix multiplications:

$$\mathbf{f}_{kk'} = \sum_{l=1}^{L}\sum_{l'=1}^{L} s_{ll'} \begin{pmatrix} f_1 & f_2 & f_3 \\ f_3 & f_1 & f_2 \\ f_2 & f_3 & f_1 \end{pmatrix} \mathbf{a}_{(k+l)(k+l')} \quad (7)$$

where $\mathbf{f}_{kk'}$ is a vectorized representation of quaternion $\hat{f}_{kk'}$, and

$$f_1 = \frac{1}{3} + \frac{2}{3}\cos\theta_{ll'}, f_2 = \frac{1}{3} - \frac{2}{3}\cos(\theta_{ll'} - \frac{\pi}{3}), f_1 = \frac{1}{3} - \frac{2}{3}\cos(\theta_{ll'} + \frac{\pi}{3}) \quad (8)$$

Proposed Quaternion CNNs

Quaternion fully-connected layers

Suppose that the input is an N-dimensional quaternion vector $\hat{A} = [\hat{a}_i] \in \mathbb{H}^N$, for $i = 1, 2, 3, ..., N$. Applying M 1D quaternion filtering kernels, *i.e.*, $\hat{W}^m = [\hat{w}_i^m] \in \mathbb{H}^M$ for $m = 1, ..., M$, we obtain an output $\hat{b} = [\hat{b}_m]$ with element

$$\hat{b}_m = \sum_{i=1}^{N} \frac{1}{s_i} \hat{w}_i^m \hat{a}_i \hat{w}_i^{m*} \tag{9}$$

Learning Quaternion CNNs
Weight initialization

For each scaling factor and each rotation factor of the j-th layer, *i.e.*, s_j and θ, and we initialize them as two uniform random variables:

$$s_j \backsim U\left[-\frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}} \quad \frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}\right], \theta \backsim U\left[-\frac{\pi}{2} \quad \frac{\pi}{2}\right] \tag{10}$$

where U represents a uniform distribution, and n_j means the dimension of the j-th layer's input.

Learning Quaternion CNNs

Backpropagation

Denote L as the real-valued loss function used to train our quaternion CNN model. $\hat{q} = q_0 + q_1 i + q_2 j + q_3 k$ and $\hat{p} = 0 + p_1 i + p_2 j + p_3 k$ are two pure quaternion variables. For the operation we perform in the QCNN, *i.e.*, $\hat{p} = \frac{1}{s}\hat{w}\hat{q}\hat{w}^*$ it can be equivalently represented by a set of matrix multiplications. So is the corresponding quaternion gradient. Particularly, we have:

$$\frac{\partial L}{\partial \mathbf{q}} = \frac{\partial L}{\partial \mathbf{p}}\frac{\partial \mathbf{p}}{\partial \mathbf{q}}, \frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \mathbf{p}}\frac{\partial \mathbf{p}}{\partial \theta}, \frac{\partial L}{\partial s} = \frac{\partial L}{\partial \mathbf{p}}\frac{\partial \mathbf{p}}{\partial s} \tag{11}$$

where $\mathrm{p} = [p_1, p_2, p_3]^T$ and $\mathrm{q} = [q_1, q_2, q_3]^T$ are vectors corresponding to $\hat{p}$ and $\hat{q}$. When p and q are arbitrary elements of feature maps and filtering kernels, corresponding to $a_{nn'}$ and $w_{ll'}$, we have

Learning Quaternion CNNs

Backpropagation

$$\frac{\partial \mathbf{p}}{\partial \mathbf{q}} = s \begin{bmatrix} f_1 & f_2 & f_3 \\ f_3 & f_1 & f_2 \\ f_2 & f_3 & f_1 \end{bmatrix}, \frac{\partial \mathbf{p}}{\partial \theta} = \begin{bmatrix} f_1' & f_2' & f_3' \\ f_3' & f_1' & f_2' \\ f_2' & f_3' & f_1' \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}, \frac{\partial \mathbf{p}}{\partial s} = \begin{bmatrix} f_1 & f_2 & f_3 \\ f_3 & f_1 & f_2 \\ f_2 & f_3 & f_1 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

where $f_i, i = 1, 2, 3$ is defined as (8) does. The matrix of f_i's is exactly same as that in (7), but the operation switches from left multiplication to right multiplication. In other words, the backward process can be explained as a rotate transform with the same axis and a reverse angle.

Learning Quaternion CNNs

Loss and activation functions

For fully-quaternion CNNs, any functions which are differentiable with respect to each part of the quaternion variables also make the quaternion chain rule hold, and thus, can be used as loss (and activation) functions. For hybrid CNNs, we select loss functions according to the category of tasks.

Experiments

We test QCNNs on two typical vision tasks: color image classification and color image denoising. These two tasks represent typical high-level and low-level vision tasks.

- Color image classification

Table 1. Experiment results in classification tasks

Model	Dataset	Test accuracy
Shallow real network	Cifar-10	0.7546
Shallow quaternion network	Cifar-10	**0.7778**
Real-valued VGG-S	102 flowers	0.7308
Quaternion VGG-S	102 flowers	**0.7695**
Quaternion VGG-S with fewer filters	102 flowers	**0.7603**

- Color image denoising

Table 2. Experiment results in denoising tasks

Model	Dataset	Test PSNR (dB)	Dataset	Test PSNR (dB)
Real-valued CNN	102 flowers	30.9792	subset of COCO	30.4900
Quaternion CNN	102 flowers	**31.3176**	subset of COCO	**30.7256**