TWiML & AI

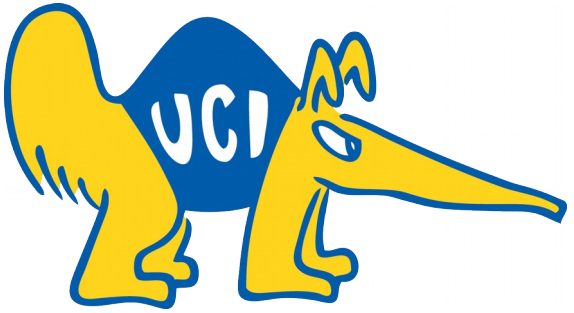
This Week in Machine Learning & AI

Topic: Deep Quaternion Networks & Quaternion Recurrent Networks

Presenter: Joseph Lopez

Who Am I?

- UCI Student:
  - Senior CS Major



- Data Science Boot camp:
  - 16 week applied course

Outline

- Part 1 – What are Quaternions?
  - The History
  - The Math
  - Their Applications
- Part 2 – Deep Quaternion Networks
  - Convolutions, Batch Norm, Weight Init
- Part 3 – Quaternion Recurrent Networks
  - Backprop through time
- Part 4 – Future Directions
  - applications
  - GHR Calculus
  - Clifford Algebras

Part 1: What Are Quaternions?

- The History:
  - Sir William Rowan Hamilton (1806 – 1865)
  - Overview Quaternions vs Vectors (1840 – Present)
- The Math of Rotation:
  - Rotation matrix and Euler Angles
  - Complex numbers and Quaternions
- Their Applications:
  - Gimble Lock
  - Applications: Graphics, Navigation, Quantum mechanics

The History

- Sir William Rowan Hamilton (1806 – 1865)
  - His Career
    - What he did
  - His Story
    - Inventing Quaternions
- Quaternions and Vector analysis
  - Grassmann
    - Exterior product and vector spaces
  - Influencers of the era
    - Maxwell, Peirce

Sir William Rowan Hamilton

- Lived his life in and near Dublin, Ireland
- made contributions to optics, dynamics, and algebra.
- Influential in dev of quantum mechanics and vector analysis
- Wanted to generalize complex numbers to describe 3d space.

His Career

- Optics
  - Ascribed characteristic functions to light rays to describe optical refraction and manipulation of light (focusing light with mirrors)
- Dynamics
  - Associated a characteristic function with any system of attracting/repelling particles
- Algebra
  - Extended complex numbers while providing conceptual basis for vector analysis

His Story

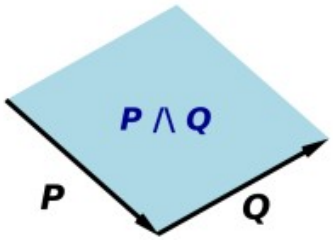
- Tried to use 3 numbers to describe 3d motion
- Was able to do it with 4 instead of 3 which was was later proven to be impossible (composition algebras)
- Real part can be scalar while the imaginary parts a 3d vector.
- Multiplication of two zero-scalared quaternions (2 3d vectors) results in a quaternion who's real component is the dot product and whose imaginary components are the cross product of those 2 3d vectors

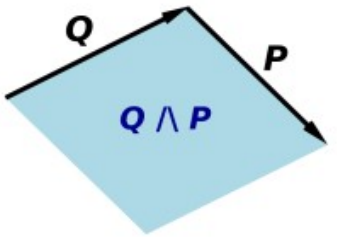
Quaternions and Vector Analysis

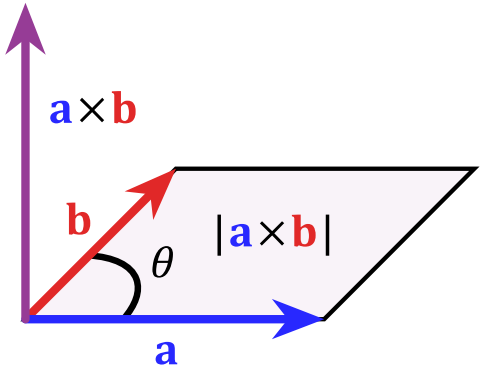
- What work was being done at the time?
- How do Quaternions relate to 3d vectors?
- What were the opinions of the influential academics of the era?
- How did Quaternions lead to the development of linear algebra?

Grassmann

- Grassmann and vectors 1840's
  - Exterior product
    - Study areas (bi-vectors),
    - volumes (tri-vectors), etc.
  - Generalized vector spaces
    - Not limited to 2 or 3d
  - Ignored for decades
    - Remained in obscurity until it was incorporated into Gibbs work on the formalization of the dot and the cross product



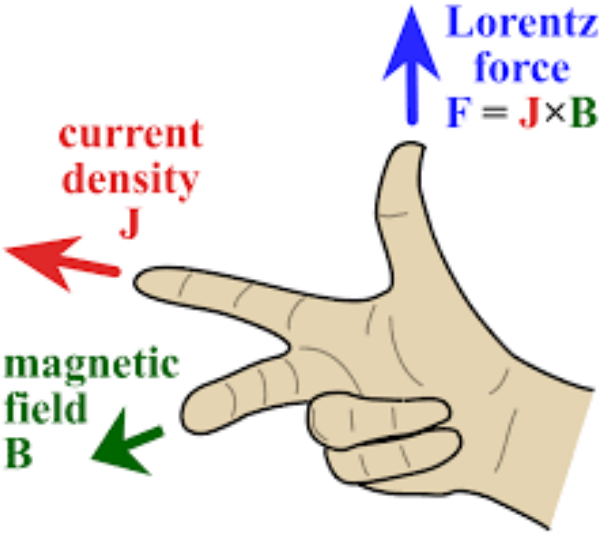




Influencers of the Era

- James Clerk Maxwell
  - Originally wrote his EM equations partly in quaternion notation bur changed to coordinate notation to make his work accessible to people unfamiliar with the notion in 1873

- Benjamin Peirce
  - Harvard Prof interested in celestial mechanics who expanded quaternions and published the first linear algebra textbook in the 1870's



The Math of Rotation

- Rotation matrix and Euler Angles
  - Rotation matrix
  - Euler Angles
- Complex and Quaternion Numbers
  - Complex algebra
  - Quaternion algebra

Rotation Matrix and Euler Angles

- Basic Linear Algebra:
  - Dot product:
    - $A * B = ||A|| \, ||B|| \cos \Theta$
  - Cross Product
    - $A \times B = ||A|| \, ||B|| \sin \Theta$

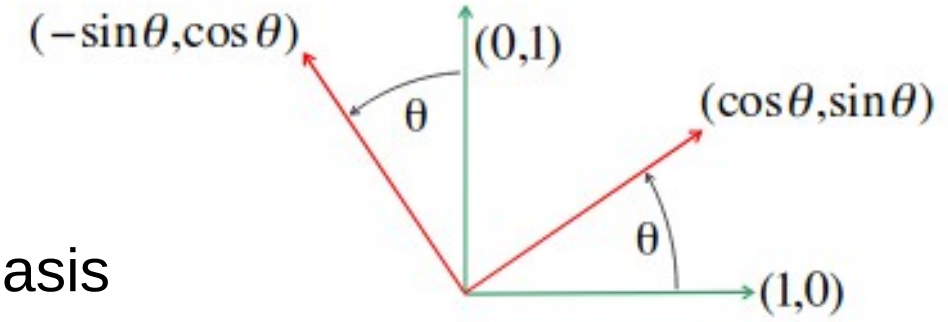
Dot & Cross Product

	b_x	b_y	b_z
a_x	Dot	Cross	Cross
a_y	Cross	Dot	Cross
a_z	Cross	Cross	Dot

All possible interactions = Similar parts + Different parts

Rotation Matrix and Euler Angles

- 2D Rotation
  - Make new basis in matrix to rotate by Θ
  - Is Orthogonal like original basis
    - Ie no sheering

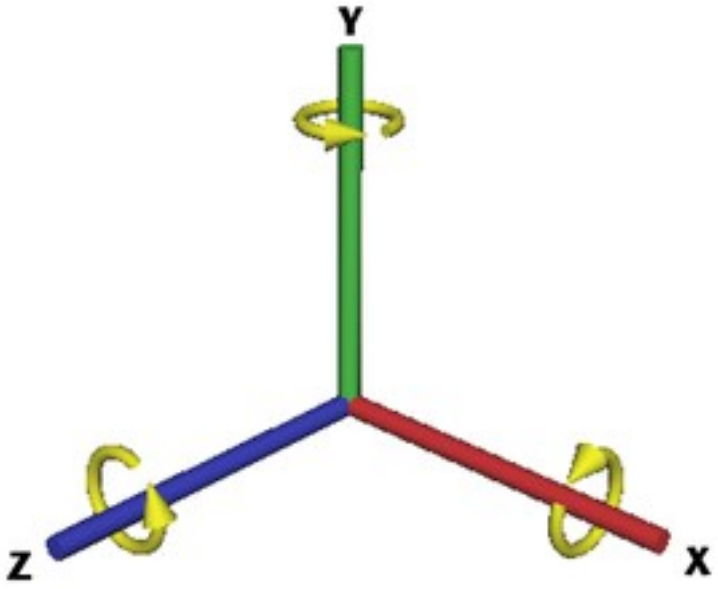


$$(x, y) \to (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$$

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Rotation Matrix and Euler Angles

- 3D rotation
  - Break space into 3 2d projections
  - Find rotation matrices for each
  - Extrapolate into 3d space

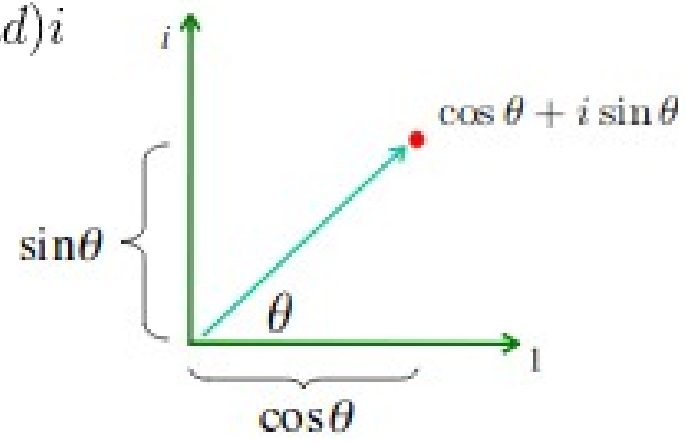


Euler rotation interpolation

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Complex and Quaternion Numbers

- Complex Numbers
  - $\mathbb{Q} = \{a + bi : a, b \in \mathbb{R}\}$ s.t. $i^2 = -1$
  - Originally invented to solve quadratic equations involving $\sqrt{-1}$
  - Add: $(a + bi) + (c + di) = (a + c) + (b + d)i$
  - Mult: $(a + bi) + (c + di) = (ac - bd) + (bc + ad)i$
  - Rot: $(a + bi) * \frac{(c + di)}{\sqrt{(c^2 + d^2)}}$
  - Euler's Formula: $e^{i\theta} = \cos\theta + i\sin\theta$
  - TIdr: mult by i rotates a vector ccw by $90°$



Complex and Quaternion Numbers

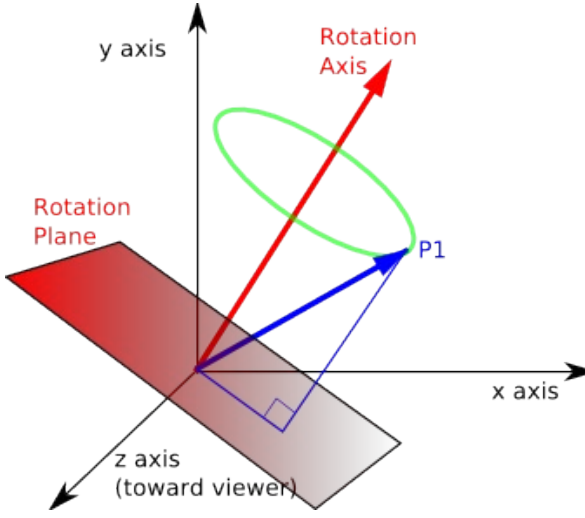
×	1	*i*	*j*	*k*
1	1	*i*	*j*	*k*
i	*i*	−1	*k*	−*j*
j	*j*	−*k*	−1	*i*
k	*k*	*j*	−*i*	−1

- Quaternion Numbers
  - $\mathbb{H} = \{a + bi + cj + dk : a, b, c, d \in \mathbb{R}\}$ s.t. $i^2 = j^2 = k^2 = ijk = -1$
  - Properties:
    - Real set to 0 results in mult being cross and dot prod
    - Real set to 1 results in mult describing 3d rotation
  - Add:

    $q_1 + q_2 = (a_0 + b_0) + (a_1 + b_1)i + (a_2 + b_2)j + (a_3 + b_3)k$
  - Non-Commutative Mult:

    $q_1 q_2 = (a_0 b_0 - \vec{a} \cdot \vec{b}) + a_0 \vec{b} i + b_0 \vec{a} j + (\vec{a} \times \vec{b})k$
  - Exp form:

    $q = e^{I \cdot \hat{n}\theta/2}$ s.t. $a_0 = cos(\theta/2)$, $\vec{a} = \hat{n}\ sin(\theta/2)$, $I = \{i, j, k\}$

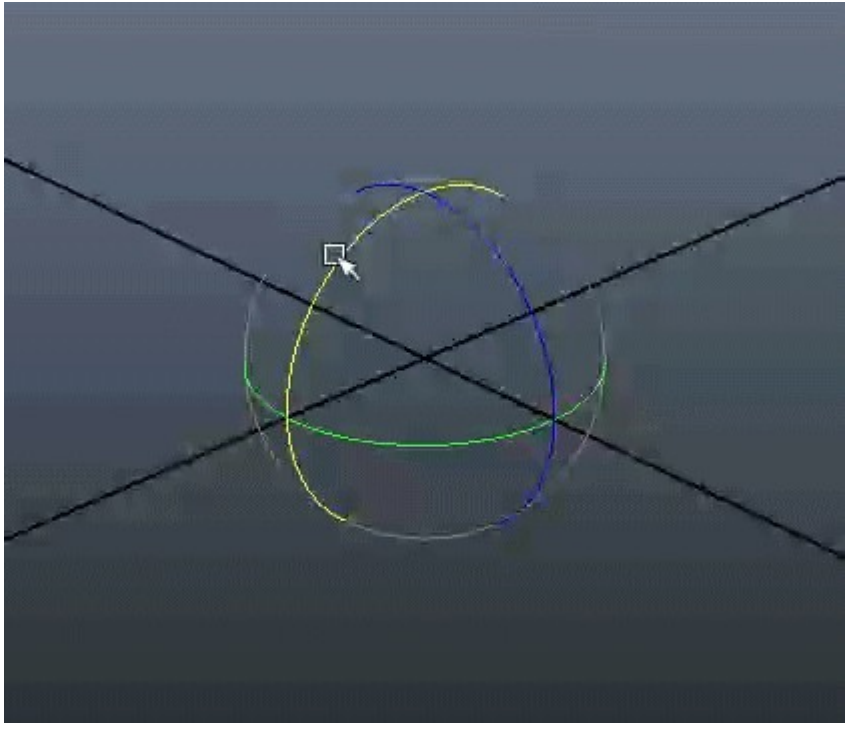


Their Applications

- Gimble Lock
  - Belt Trick, Gyroscope
- Applications
  - Computer Graphics
  - Aerospace Guidance
  - Quantum Mechanics

Gimble Lock

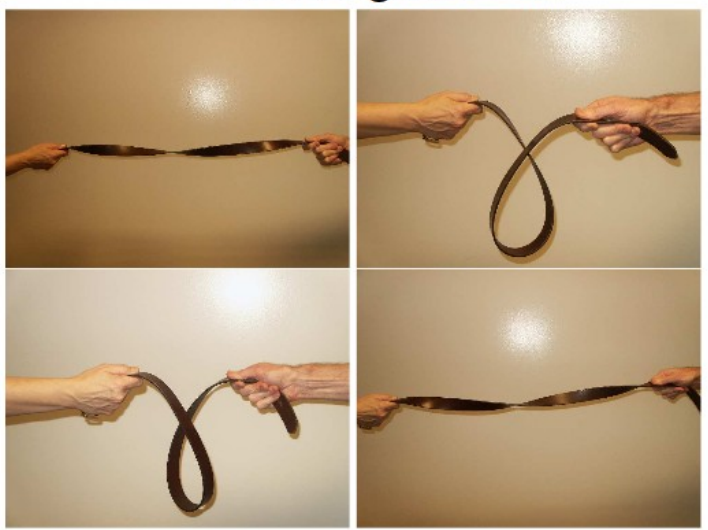
- The loss of a degree of freedom due to a Cartesian based orientation control sequence.
- In 3d if two axes align, you can’t tell which of the two aligned axes an orientation is coming from.



The Belt Trick

- Euler angles use 3 angles each representing up to 360 degrees of rotation while quaternions use 1 angle for 720 degrees of rotation.

360 Degree Belt



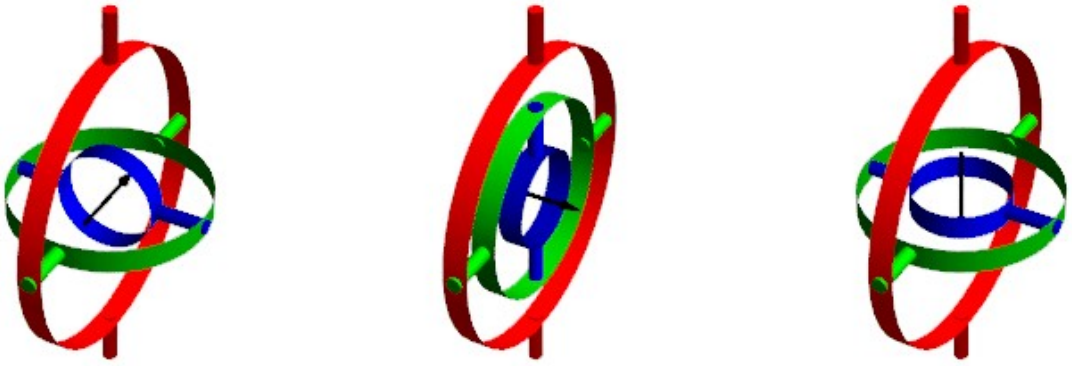
360 twist: stays twisted, can change DIRECTION!

720 Degree Belt

720 twist: CAN FLATTEN OUT WHOLE BELT!

Gyroscope

- A Gimble can be thought of as a fixed axis of rotation. With Euler angles, to represent rotations you would decide on an ordering of rotation matrices which would be used and and updated to express changes in orientation. If Gimble lock occurs, it is impossible to fix the axes without an external reference.



Applications

- Computer Graphics:
  - Unity hard codes rotations in quaternions, giving the user the option to use them or Euler angles (which can be simulated with said quaternion representation). Allows for smooth rotations.
- Aerospace Guidance
  - More compact representation (4 vs 9 nums) results in lower computation error, more resilient against perturbations (an slightly off quaternion still represents rotation while a slightly off matrix may not be orthogonal anymore). All good for guidance and navigation systems.

Applications

- Quantum Mechanics
  - Objects related to quaternions arise from the solution of the Dirac equation for the electron where the algebra's non-communative property is essential.
    - To rotate a particle you rotate its wave via complex nums
    - To rotate an electron quaternions can be used
  - The quaternions are closely related to various spin matricies of quantum mechanics

Part 2: Deep Quaternion Networks

Chase J. Gaudet
School of Computing & Informatics
University of Lousiana at Lafayette
Lafayette, USA
cjg7182@louisiana.edu

Anthony S. Maida
School of Computing & Informatics
University of Lousiana at Lafayette
Lafayette, USA
maida@louisiana.edu

Deep Quaternion Networks

- Abstract's TLDR:
  - Provide components to build a deep quaternion net
  - Develop novel quaternion convolution, quaternion weight initialization, and quaternion batch-norm
  - Tested on classification and segmentation task
  - State improved convergence while having fewer parameters

Deep Quaternion Networks

- Introduction
- Motivation and Related Work
- Quaternion Network Components
- Experimental Results

Introduction

- Batch Norm: Regularize net & provide faster/stable training by standardizing activations according to minibatch stats
- Shortcut Paths: connecting later layers w/ earlier layers, stronger backprop, help fight against vanishing gradients.
- Complex RNNs faster/more robust learning
- Work has been done on complex nets (and quaternion signal processing

Motivation and Related Work

- The Ability of quaternions to effectively represent spatial transformations and analyze multidimensional signals makes them promising. Quaternions can be represented as complex groups providing for more robust and efficient mechanisms than can real or complex representations.
- PoseNet: Target output was a quaternion to recover 6-DOF camera pose from an RGB image.
- Quaternion signal processing: Signal in phase of an image can be sufficiently recovered magnitude (real component).
- Quaternion Convolutional filters: real as gray scale, RGB as imaginaries w/ masks

Quaternion Network Components

- Quaternion representation
  - Intro to Hamilton and Quaternions
  - Quaternion arithmetic using reals
    - Real Mapping is used for paper but others like $\mathbb{H} \mapsto M(2, \mathbb{C})$ exist
  - 2D Convolution layer
    - Given N feature maps, divide into 4 separate components and give to each quaternion channel

$$\mathbb{H} \mapsto M(4, \mathbb{R})$$

$$\begin{bmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{bmatrix} = a \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + b \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} + c \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} + d \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Quaternion Network Components

- Quaternion Differentiability
  - Provide quaternion chain rule for backprop for real valued functions
    - L as loss func, quaternions q = a+bi+cj+dk and g = m + ni+oj + pk

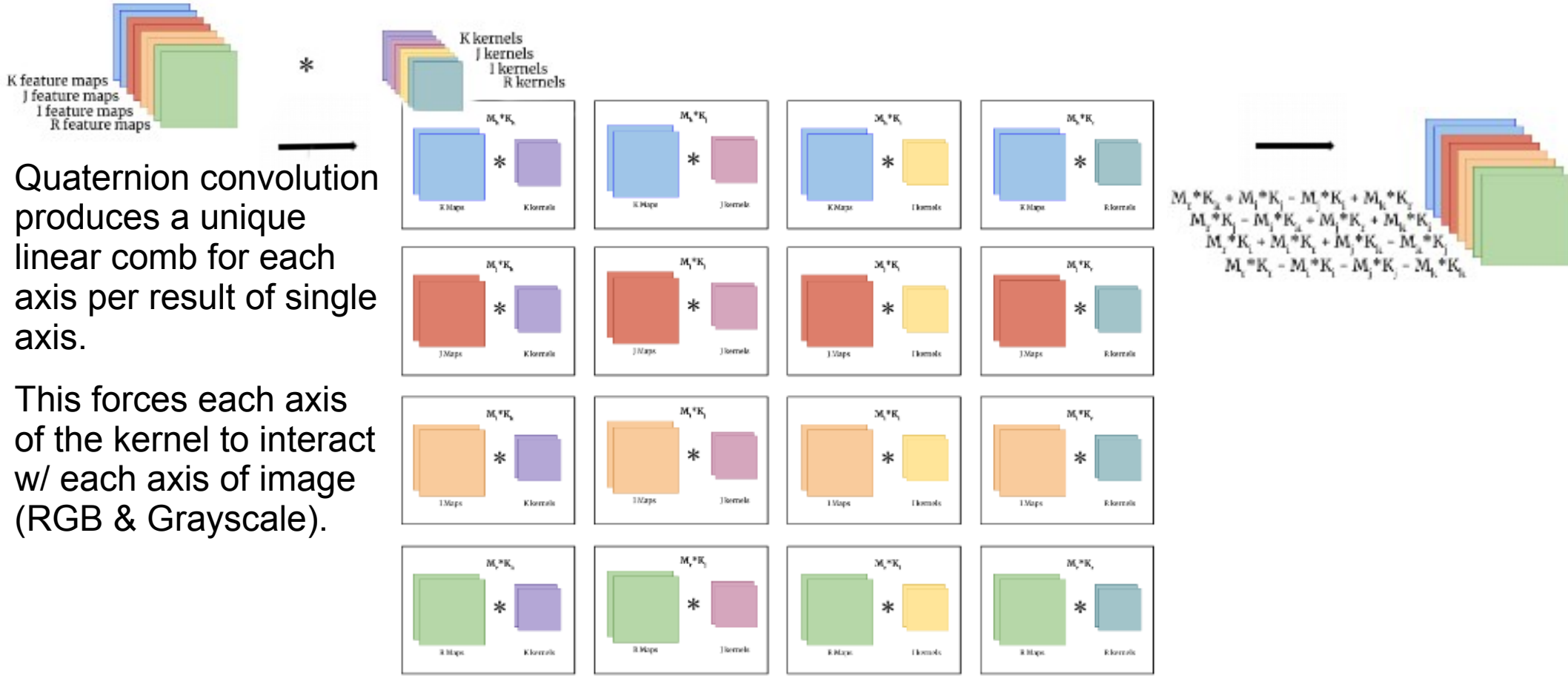
$$\nabla_L(q) = \frac{\partial L}{\partial g} = \mathbb{R}(\nabla_L(q)) \left(\frac{\partial a}{\partial m} + i \frac{\partial a}{\partial n} + j \frac{\partial a}{\partial o} + k \frac{\partial a}{\partial p} \right)$$
$$+ \mathbb{I}(\nabla_L(q)) \left(\frac{\partial b}{\partial m} + i \frac{\partial b}{\partial n} + j \frac{\partial b}{\partial o} + k \frac{\partial b}{\partial p} \right)$$
$$+ \mathbb{J}(\nabla_L(q)) \left(\frac{\partial c}{\partial m} + i \frac{\partial c}{\partial n} + j \frac{\partial c}{\partial o} + k \frac{\partial c}{\partial p} \right)$$
$$+ \mathbb{K}(\nabla_L(q)) \left(\frac{\partial d}{\partial m} + i \frac{\partial d}{\partial n} + j \frac{\partial d}{\partial o} + k \frac{\partial d}{\partial p} \right)$$

Quaternion Network Components

- Quaternion Convolution
  - Convolving a quaternion filter matrix **W = A** + **B**i + **C**j + **D**k by a quaternion vector **h = w** +**x**i +**y**j +**z**k where **A**, **B**, **C**, **D** are real valued matrices and **w**,**x**,**y**,**z** real valued vectors results in the following once reduced into matrix form.

$$\begin{bmatrix} \mathscr{R}(\mathbf{W} * \mathbf{h}) \\ \mathscr{I}(\mathbf{W} * \mathbf{h}) \\ \mathscr{J}(\mathbf{W} * \mathbf{h}) \\ \mathscr{K}(\mathbf{W} * \mathbf{h}) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & -\mathbf{B} & -\mathbf{C} & -\mathbf{D} \\ \mathbf{B} & \mathbf{A} & -\mathbf{D} & \mathbf{C} \\ \mathbf{C} & \mathbf{D} & \mathbf{A} & -\mathbf{B} \\ \mathbf{D} & -\mathbf{C} & \mathbf{B} & \mathbf{A} \end{bmatrix} * \begin{bmatrix} \mathbf{w} \\ \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{bmatrix}$$

Quaternion Neural Networks



Quaternion convolution produces a unique linear comb for each axis per result of single axis.

This forces each axis of the kernel to interact w/ each axis of image (RGB & Grayscale).

Quaternion Neural Networks

- Quaternion Convolution
  - Similar to combo of standard & depthwise separable convolutions
    - Depthwise separable convolutions: flat kernel applied separately to each feature map (RGB & Grayscale)
      - 1x1 convolution applied to them to produce linear interaction of feature maps
  - Network’s reuse of these filters on every axis may help extract more info across channels
  - Since Quaternions can be thought of as a vector, kernels and feature maps of quaternion convolution can be thought of as vectors as well

Quaternion Neural Networks

- Quaternion Batch Norm
  - No easy normalization (mean of 0, std of 1) exists for complex/hypercomplex nums
  - Whitening approach used as there's no easy way to invert a M(4,R) matrix
    - Whitening decorrelates the input dimensions (cov is Identity matrix)
    - Cholesky decomposition of cov matrix used instead to find **W**
      - Algo to decompose a symmetric matrix into a lower and an upper triangular matrix A = L * L'
  - Whitening accomplished by $\tilde{x} = \mathbf{W}(\mathbf{x} - \mathbb{E}[\mathbf{x}])$ where **W** is found by cholesky decomp of cov matrix **V**

Quaternion Neural Networks

- Quaternion Batch Norm
  - Cov matrix **V** given by where each V is cov between real and imaginaries
  - Real batch norm has 2 learned params
    - Shift param β which is quaternion valued in our case
    - Scale param γ which is a symmetric matrix according to cov matrix

$$\mathbf{V} = \begin{bmatrix} V_{rr} & V_{ri} & V_{rj} & V_{rk} \\ V_{ir} & V_{ii} & V_{ij} & V_{ik} \\ V_{jr} & V_{ji} & V_{jj} & V_{jk} \\ V_{kr} & V_{ki} & V_{kj} & V_{kk} \end{bmatrix}$$

$$\gamma = \begin{pmatrix} \gamma_{rr} & \gamma_{ri} & \gamma_{rj} & \gamma_{rk} \\ \gamma_{ri} & \gamma_{ii} & \gamma_{ij} & \gamma_{ik} \\ \gamma_{rj} & \gamma_{ij} & \gamma_{jj} & \gamma_{jk} \\ \gamma_{rk} & \gamma_{ik} & \gamma_{jk} & \gamma_{kk} \end{pmatrix}$$

Quaternion Neural Networks

- Quaternion Batch Norm
  - Since cov($\tilde{x}$) = Identity matrix diagonal of γ set to $1/\sqrt{4}$ to obtain a modulus of 1 for variance of normalized value
  - Quaternion batch norm defined as

$$\mathrm{BN}(\tilde{\mathbf{x}}) = \gamma\tilde{\mathbf{x}} + \beta$$

Quaternion Neural Networks

- Quaternion Weight Initialization
  - Begin by finding var of a quaternion weight

$$W = |W|e^{(\cos\phi_1 i + \cos\phi_2 j + \cos\phi_3 k)\theta}$$
$$= \mathscr{R}\{W\} + \mathscr{I}\{W\} + \mathscr{J}\{W\} + \mathscr{K}\{W\}$$

  - Where |W| is magnitude, Θ and Φ angle args and

$$\cos^2\phi_1 + \cos^2\phi_2 + \cos^2\phi_3 = 1$$

  - Variance is defined by $\mathrm{Var}(W) = \mathbb{E}[|W|^2] - (\mathbb{E}[W])^2$
    - Since W is symmetric about 0, second term is 0.

Quaternion Neural Networks

- Quaternion Weight Initialization
  - No easy way to calculate Var(W)
    - Use integration on |W| $\mathbb{E}[|W|^2] = \int_{-\infty}^{\infty} x^2 f(x)\, dx = 4\sigma^2$
      - f is pdf for the 4DOF normal distribution
      - Express Var(W) by single param σ
  - Follow setup from another paper
    - $\mathrm{Var}(W) = 2/(n_{in}+n_{out})$. Set equal to $4\sigma^2$, solve for $\sigma = 1/\sqrt{2(n_{in}+n_{out})}$
    - In case if RELUs: $\mathrm{Var}(W) = 2/n_{in}$ solving gives $\sigma = 1/\sqrt{2n_{in}}$

Quaternion Neural Networks

- Quaternion Weight Initialization
  - Can init |W| according to pdf parameterized by σ
  - Angle components Φ and Θ initialized according to normal
  - distribution between – π and π to constrain Φ

Quaternion Neural Networks

- Experimental Results
  - Architectures used
    - 3 stages of repeating residual blocks
    - More efficient way to do quaternion nets for image input
  - Image classification
    - using CIFAR-10 and CIFAR-100 benchmarks
    - 32x32 colored images of 10 & 100 classes
  - Image segmentation
    - using KITTI Road Estimation benchmark
    - Large colored images of varying size from driver’s POV
      - Each pixel has corresponding integer label saying if it is part of a road or not

Quaternion Neural Networks

- Architectures
  - Additional residual block after input to learn complex/quaternion components $BN \rightarrow ReLU \rightarrow Conv$
    - Have one of these blocks per imaginary component concatenated w/ original
      - For quaternion net, if input is an RGB image, grayscale and RGB can be 4 axes instead
  - All architectures had 3 stages of these repeating residual blocks

    $BN \rightarrow ReLU \rightarrow Conv \rightarrow BN \rightarrow ReLU \rightarrow Conv$

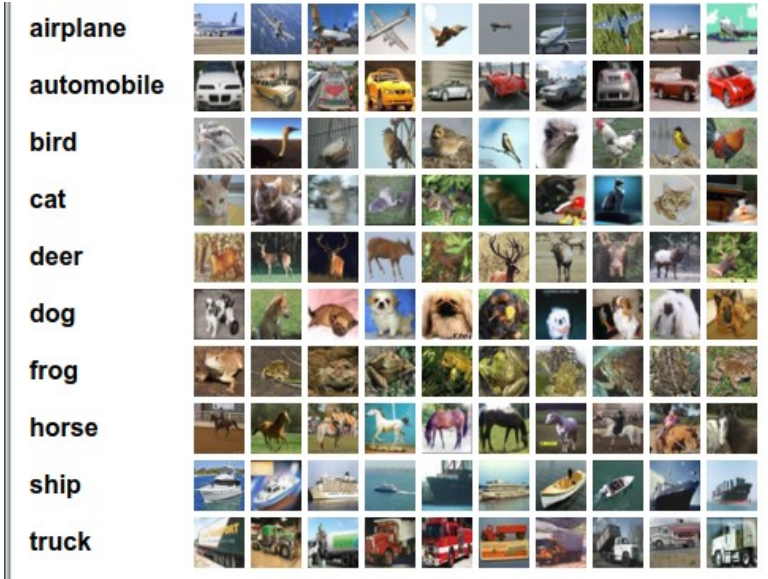
    where images are downsized via a strided convolution

Deep Quaternion Nets

- Image Classification
  - CIFAR-10/CIFAR-100
    - Images w/ 10 & 100 labeled classes respectively
  - Experiment
    - Shallow and deep real, complex, and Quaternion nets
  - Results
    - Longer to train, outperform w/ less params

Deep Quaternion Networks

- CIFAR-10
  - 60,000 32x32 images of 10 classes
    - 600 images of each class (500 train, 100 test)
- CIFAR-100
  - CIFAR-10 except 100 classes
    - 20 super classes containing 5 classes each



Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates

Deep Quaternion Networks

- Experiment
  - Network types: real, complex, quaternion
    - One shallow and one deep net for each type on both CIFAR benchmarks
      - Shallow: stages contain 2, 1 and 1 residual blocks
      - Deep: stages contain 10, 9, and 9 residual blocks
      - Each stage doubles previous's num of convolutions
        - Ex: real had 34 kernels, 64, then 124 in last stage
  - Parameter budget:
    - Convolutional filters
      - 2x less for complex
      - 4x less for quaternions
  - Training:
    - SGD w/ Nestrov momentum of .9
    - Gradients clipped to 1
    - Custom training schedule (same as in paper on complex nets for comparison)
      - .01 for 10 epochs, .1 for 11-100 epochs, cut by 10 at 120 and 150 epochs

Deep Quaternion Networks

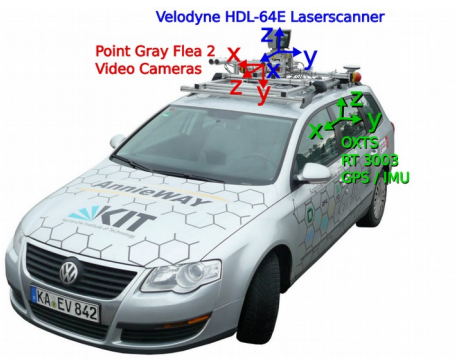
- Results:
  - Quaternion outperformed real & complex nets on both data sets w/ fewer params
  - Quaternion models took ~50 % longer to train due to computation heavy quaternion batch norm

Architecture	Params	CIFAR-10	CIFAR-100
Shallow Real	508,932	6.82	32.02
Shallow Complex	257,412	6.91	31.65
Shallow Quaternion	133,560	**6.77**	**30.59**
Deep Real	3,619,844	6.37	28.07
Deep Complex	1,823,620	5.60	27.09
Deep Quaternion	932,792	**5.44**	**26.01**

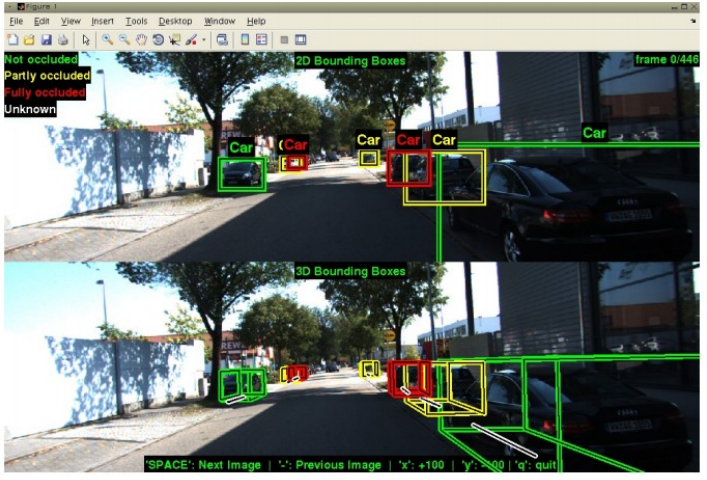
TABLE I

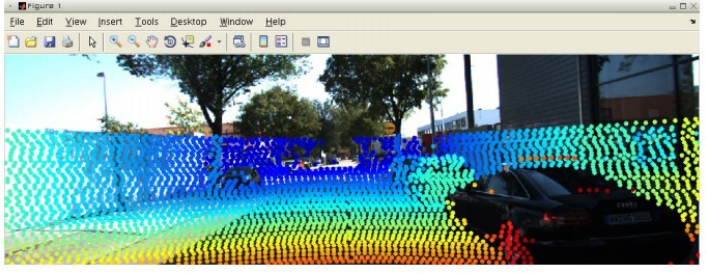
CLASSIFICATION ERROR ON CIFAR-10 AND CIFAR-100. PARAMS IS THE TOTAL NUMBER OF PARAMETERS.

Deep Quaternion Networks



- Image Segmentation
  - KITTI Data set
    - Designed or 3d object detection & tracking, stereo, optical flow, and visual odometry
    - Real world CV benchmark from driver's POV
    - Images labeled w/ integer heatmap saying if a pixel is part of a road or not
    - Large images (~1200x375)





Deep Quaternion Networks

- Experiment
  - Used only shallow nets like in classification experiment due to resource limitations.
    - (2, 1, 1) residual blocks at 3 stages
    - No strided convolutions
    - No global average pooling/fully connected layer at end
    - 1x1 convolution w/ sigmoid activation at end to produce heatmap similar to input
  - Same training procedure but different learning rate scheduler
    - .01 for 1st 10 epochs, .1 for 11-50 epochs, divide by 10 at 100 and 150 epochs

Deep Quaternion Networks

- Results
  - Used Intersection Over Union (IOU) metric
  - Quaternions outperformed other 2 by large margin than classification task w/ smaller param count.

IOU is Jaccard index:
complement can be used as distance metric for dissimilarity between sample sets

Architecture	Params	KITTI
Real	507,029	0.747
Complex	254,037	0.769
Quaternion	128,701	**0.827**

TABLE II
IOU ON KITTI ROAD ESTIMATION BENCHMARK.

Deep Quaternion Networks

- Conclusions
  - Quaternion Recurrent Neural Networks
    - Reduce number of free params by 3x to reach same/better results
    - Novel backprop through time algo
  - Learning Algo’s in QNN using GHR calculus
    - Optimization on Quaterions themselves instead of on cov matrix
  - Clifford Algebras
    - Neural computation in geometric algebras that aren’t quaternions