Quaternion Neural Networks for Sequence Classification

Follow this and additional works at: https://ecommons.udayton.edu/stander_posters

Recommended Citation
"Quaternion Neural Networks for Sequence Classification" (2019). Stander Symposium Posters. 1447. https://ecommons.udayton.edu/stander_posters/1447

This Book is brought to you for free and open access by the Stander Symposium at eCommons. It has been accepted for inclusion in Stander Symposium Posters by an authorized administrator of eCommons. For more information, please contact frice1@udayton.edu, mschlangen1@udayton.edu.
Quaternion Neural Networks for Sequence Classification

Introduction

• Typically Machine Learning sequence processing has been done using gated cells such as the Long Short-Term Memory Cell (LSTM)
• Recent work has shown that a Causal, Dilated, Fully convolutional network design called the Temporal Convolutional Network (TCN) is capable of meeting and outperforming gated cell units on sequences of data
• Other work has shown that by using quaternion representations of weights in Deep learning Networks it is possible to reduce the number of learned parameters by 75% for about the same performance

Problem Statement

• Can we combine the Temporal Convolutional Network design with quaternion convolutions to produce the same results for sequence data?

Temporal Convolutional Network

• A network that takes in a sequence of data and produces an output sequences of the same length
• Each layer of the network is made up of a residual block that performs 1D, dilated causal, convolutions on the layer below it
• The dilations allow for the networks receptive field to grow as the network passes data through it. At the final layer the receptive field will cover the entire input sequence

Quatcron Convolutions

A quaternion convolution is done by taking a quaternion filter matrix notated as:
\[ W = A + iB + jC + kD \]
And convolving it a quaternion vector:
\[ h = w + ix + jy + kz \]
Can be reduced down to the form:
\[
\begin{bmatrix}
    \Re(W*h) \\
    \Im(W*h) \\
    \j(W*h) \\
    \k(W*h)
\end{bmatrix} =
\begin{bmatrix}
    A & -B & -C & -D \\
    B & A & -D & C \\
    C & D & A & -B \\
    D & -C & B & A
\end{bmatrix} *
\begin{bmatrix}
    w \\
    x \\
    y \\
    z
\end{bmatrix}
\]
For a 1D quaternion Convolution we convolve each row of the kernel matrix with each input channel and then sum them to make sure that each channel is convolved with all possible unique axis of the kernel matrix. Each convolution step in the QTCN is 4 convolutions and a summation.

Results

<table>
<thead>
<tr>
<th>Network Type</th>
<th># of Learned Parameters</th>
<th>Validation Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>~1 M</td>
<td>71.020%</td>
</tr>
<tr>
<td>TCN</td>
<td>~440k</td>
<td>77.820%</td>
</tr>
<tr>
<td>QTCN</td>
<td>~112K</td>
<td>75.820%</td>
</tr>
</tbody>
</table>

Methodology

• We take the 10 class RGB dataset CIFAR10 and flatten it to form the “Sequential CIFAR10” dataset
• We add a grayscale version of the image to the RGB images to produce 4 channels for the quaternion convolutions
• The LSTM is given a hidden state of 512
• The TCN and QTCN are 8 layers of 64 filters with a kernel size of 10