Deep Quaternion Neural Networks for 3D Sound Source Localization and Detection

Project of Neural Network Course

Sveva Pepe 1743997
Marco Pennese 1749223
Claudia Medaglia 1758095

1 Introduction

Sound source localization is a fundamental task, expecially in revembrant and multiple sources envoirments.
In this project, we work with 3D audio sounds captured by first-order Ambisonic microphone and these sounds are then represented by spherical harmonics decomposition in the quaternion domain.
The aim of the project is to detect the temporal activities of a known set of sound event classes and to further locate them in the space using quaternion-valued data processing, in particular we focus on the sound event localization and detection.
In order to do this, we use a given Quaternion Convolutional Neural Network with the addition of some recurrent layers (QCNN) for the joint 3D sound event localization and detection (SELD) task.

2 Quaternion domain

What are quaternions, and their connection with 3D audio recorded with Ambisonic. Enter the mathematical formula on the composition (that of the real part + imaginary part). A minimum of considerations on active and reactive intensity, in particular the role of active and reactive intensity in DOA.
Formulas of the two "input features" with quaternions.

3 Network Structure

The model receives as input the quaternions, from which it extracts the spectrogram in terms of magnitude and phase components using a Hamming window.
The network is a Quaternion Convolutional Recurrent Neural Network (QCRNN). In particular, we have three convolutional layers based on quaternions (QCNN), two recurrent layers (QRNN) and finally two parallel outputs, both are composed of two fully-connected layers, which obviously differ in their activation functions and size that receive input from previous layers.
The QCNN layers are composed of P filter kernels with size $3x3x8$.
The three convolutional layers (QCNN) consist of 3 stages: Convolutional, Detector and Pooling.
The first stage consists of the convolutional process to the inputs. Since these are quaternions, the convolutional process consists of Hamilton's product:

$$\begin{aligned} W \otimes x = &(W_w x_w - W_x x_x - W_y x_y - W_z x_z) \\ &+(W_w x_x + W_x x_w + W_y x_z - W_z x_y)\hat{\imath} \\ &+(W_w x_y - W_x x_z + W_y x_w + W_z x_x)\hat{\jmath} \\ &+(W_w x_z + W_x x_y - W_y x_x - W_z x_w)\hat{\imath} \end{aligned}$$

where x is the vector of the quaternions taken as input and W a generic quaternion filter matrix.
Hamilton product allows quaternion neural networks tocapture internal latent relations within the features of a quaternion.
After it is applied the Batch Normalization, a technique for improving the speed, performance, and stability of artificial neural networks. It is used to normalize the input layer by adjusting and scaling the activations. Also, batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers.
The second phase concerns the choice of the activation function, which in the case of the three quaternion

convolutional layer is the ReLU.[1]
For a generic quaternion dense layer we have:

$$y = \alpha(W \otimes x + b)$$

where y is the output of the layer, b is the bias offset and α is quaternion activation function.
Indeed, $\alpha(q) = f(q_w) + f(q_x) + f(q_y) + f(q_z)$, q is a generic quaternion and f is rectified linear unit (ReLU) activation function. As we can see, The ReLU is applied to both the real and imaginary part of the quaternion.

The last stage is the one related to Pooling. Pooling decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model. In our network we use MaxPooling. MaxPooling is done by applying a max filter to non-overlapping subregions of the initial representation.
Max-pooling is applied along the frequency axis for dimensionality reduction while preserving the sequence length T, we obtain in the end that the output of the 3 QCN has dimension T x 8P and fed to bidirectional QRNN layers to better catch the time progress of the input signals.

QRNN are recurring neural networks based on quaternions. The peculiarity of recurrent neural networks is that they differ from feedforward nets because they include a feedback loop, whereby output from step n-1 is fed back to the net to affect the outcome of step n, and so forth for each subsequent step.
In these QRNN Q nodes of quaternion gated recurrent units (QGRU) are used in each layer and as a function of activation a hyperbolic tangent (tanh).

The output of the recurrent layer is shared between two fully-connected layer branches each producing the SED as multi-class multilabel classification and DOA as multi-output regression; together producing the SELD output.

4 Dataset

The dataset, what it is made of, and how it was divided by us. How many files each subfolder is made up of.

5 Metrics

The metrics used, therefore the average, SELD SCORE, the confidence interval and I do not know whether to consider the F1 score as well. Obviously with formulas

6 Experiments

Our experiments, then the changes we made, the results that come to us, the various graphs. Let's see what comes out of it, and in case I would make comparisons with the results that have been obtained from the various papers that have provided us.

7 Conclusion

The conclusions regarding the project carried out and you have had results based on the metrics that we used.

[1]ReLU stands for rectified linear unit, and is a type of activation function. Mathematically, it is defined as $y = \max(0, x)$ where y is the output and x is the input.