Full-learning quaternion convolutional neural networks and cross-entropy-based confluence of differently represented data for PolSAR land classification

Yuya Matsumoto

1University of Tokyo

October 30, 2023

Abstract

ALOS2 Dataset
OdilSceneId="SARD000000088012-00036-019-001"
OdilSiteDateTime="PROCESS:JAPAN-JAXA-ALOS2-EICS 20151207 072121"
ScsSceneID="ALOS2066310850-150815"

ALOS2 Dataset
OdilSceneId="SARD000000088012-00036-044-002"
OdilSiteDateTime="PROCESS:JAPAN-JAXA-ALOS2-EICS 20210113 054828"
ScsSceneID="ALOS2066310860-150815"
Full-learning quaternion convolutional neural networks and cross-entropy-based confluence of differently represented data for PolSAR land classification

Yuya Matsumoto, Ryo Natsuaki Member, IEEE, and Akira Hirose, Fellow, IEEE

Abstract—This paper proposes two methods in adaptive PolSAR land classification. The first proposal is full-learning quaternion convolutional neural networks (QCNNs). QCNNs can learn the relationship between the components of input vectors by quaternion rotations unlike conventional real-valued convolutional neural networks (RVCNNs). They have been employed in some image classification tasks including PolSAR images. However, the existing QCNNs fix their quaternion-rotation axes, so that they cannot realize arbitrary quaternion rotations. Therefore, the degree of freedom (DoF) in the fixed-axis quaternion is two while a quaternion holds DoF of four intrinsically. Our proposed full-learning QCNN is a novel neural network which updates all the four elements of respective quaternion neural weights including its convolutional kernels by backpropagation. The novel QCNN has the maximum DoF so that it takes advantage of quaternionic operation. The second proposal is a method of combining multiple-network classification results for different features. Classification results for PolSAR images largely depend on input features. Therefore, merging multiple results for different features improves the final classification results by compensating for the shortcomings of respective feature sets. In our experiments, we employ the following two features, namely, PolSAR pseudo-color features and normalized Stokes vectors. After the classification for the respective features, the classification results are merged based on the cross-entropy loss. We compare three neural networks, namely, RVCNNs, fixed-axis QCNNs and our proposed full-learning QCNNs. Our proposed QCNNs show the best classification results for respective features. We also show that the merging process highly improves the final classification results measured by F-scores.

Index Terms—Polarimetric synthetic aperture radar (PolSAR), convolutional neural network (CNN), quaternion neural network (QNN)

I. INTRODUCTION

Polarimetric synthetic aperture radar (PolSAR) systems obtain more information by using multiple polarizations than by using a single polarization [1]. Land classification is one of the most important PolSAR applications. There are various methods based on, e.g., entropy and alpha angle [2], matrix decomposition [3]–[5], wishart distribution [6], [7], and traditional machine learning methods such as support vector machine (SVM) [8], [9] and random forest [10], [11]. Recently, neural-network methods have been actively studied and they have shown much better performance than the conventional methods [12]–[15].

There are unique difficulties for PolSAR image classification, unlike general image classification, such as selection of optimal features, less labeled data, and speckle noise [16]. Hence, some unique neural networks for PolSAR land classification have been proposed.

Quaternion neural networks (QNNs) [17] have been employed for PolSAR land classification [18]–[20]. In QNN methods, a three dimensional vector is regarded as a single quaternion. In other words, the three components of the vector are assigned to the three imaginary parts of a quaternion. QNNs restrict the linear transformation to quaternion rotation, that is, orthogonal transformation. Therefore, the input features do not degenerate. When we pay attention to a single neuron, we find that it maps them to a space of the same dimension with less parameters than three real-valued neurons. Then, the QNNs have shown higher accuracy than the conventional PolSAR classification methods.

Quaternion convolutional neural networks (QCNNs) are employed for visible RGB-color image [21] and PolSAR image [22] classification. The QCNNs have kernels consisting of quaternion neurons to learn spatial texture. In general, real-valued convolutional neural networks (RVCNNs) simply sum up the components of the input vectors in the convolutional processing, so that they ignore the relationship between the components. On the other hand, the QCNNs perform quaternionic rotation, that is, orthogonal transformation, on the respective input vectors in the convolutional processing. Hence, in the QCNNs, the components of input vectors do not degenerate, so that the QCNNs can learn the relationship between the components.

However, there are two problems in the QCNN methods. First, the existing QCNNs fix the rotation axes of their quaternion-weight transform in neurons to reduce the degree of freedom [21], [22]. Hence the QCNNs cannot represent arbitrary rotations, and lose the expression ability. Second, the only PolSAR pseudo-color features are employed as input features [22]. With only intensity features such as PolSAR pseudo-color features, it is difficult to discriminate classes...
with low back-scattering such as sea and grass, which is to be
discussed later.

This paper proposes the following two methods to address
the above problems. First, we propose a full-learning QCNN
which updates all four parameters of its quaternion weights
of neurons, even in its convolutional kernels, by backpropagation
without fixing the rotation axes. The full-learning QCNNs can
perform arbitrary quaternion rotation on the input vectors.
In other words, our proposed QCNNs have the maximum
degree of freedom to take advantage of quaternionic rotations
unlike the existing fixed-axis QCNN. We propose the detailed
dynamics in this paper.

Second, we propose a method of combining two classi-
formations for different features. It is very effective to
merge classification results for different features since different
features are suitable for representing diverse nature. In our
experiments, we employ two sets of features, namely, PolSAR
pseudo-color images and Stokes vectors normalized by their
total power. The former are derived by assigning the intensities
of scattering coefficients to red, green, and blue components
based on the Pauli decomposition. These intensity features in-
clude fine textures of a PolSAR image. By learning the spatial
textures, neural networks can distinguish between classes such
as town and forest, which have so similar scattering charac-
teristics in pixels that they are often misclassified each other.

Another set of features, the Stokes vectors normalized by their
total power, have the degree of polarization (DoP) representing
the state of a partially polarized wave. Most scattered waves
are partially polarized, and the DoP is significant information
for land classification. It is very effective to learn the DoP in
discriminating between classes with low backscattering such
as sea and grass, on which we focus in this paper. The cross-
entropy suggests which set of features is more effective in the
pixel-wise classification result.

In the experiments in this paper, we compare three neu-
networks, namely, RVCNNs, fixed-axis QCNNs and our
proposed QCNN. As experimental results, our proposed QC-
NNs show the best classification performance by learning the
relationship between the components of input vectors with ar-
bitrary quaternion rotations. The neural networks learning the
PolSAR pseudo-color images succeed in discriminating town
and forest, while the neural networks learning the normal-
ized Stokes vectors succeed in sea and grass discrimination.
Merging the two classification results for different features compensates for the shortcomings of respective features to
improve the final classification results measured by F-scores.

This paper is organized as follows. Section II presents
the theory of the full-learning QCNN. Section III describes
the details of the experiments. The experimental results are
explained in Section IV. Section V concludes this paper.

II. QUATERNION CONVOLUTIONAL NEURAL
NETWORKS

A. Quaternion arithmetic

This section introduces the detailed theory of a full-learning quaternion convolutional neural network. First of all, we
describe the expressions of various operations of quaternions. A quaternion is expressed as

$$
\begin{align*}
x &= x^e + x^i i + x^j j + x^k k \\
&= (x^e, x^i, x^j, x^k) \\
&= \begin{bmatrix}
x^e \\
x^i \\
x^j \\
x^k
\end{bmatrix}
\end{align*}
$$

where \(i, j, \text{ and } k\) stand for the three imaginary units of a quaternion. They satisfy the following Hamilton rules:

$$
\begin{align*}
i^2 &= j^2 = k^2 = -1 \\
i j &= -i j = k, j k = -k j = i, k i &= -i k = j
\end{align*}
$$

The operators between quaternions \(p = (p^e, p^i, p^j, p^k)\) and \(q = (q^e, q^i, q^j, q^k) = (q^e, q^v)\) are defined as

Addition and subtraction:

$$
p \pm q = (p^e \pm q^e, v_p \pm v_q)
$$

Outer product:

$$
p \otimes q = (p^e q^e - v_p \cdot v_q, p^e v_q + q^e v_p + v_p \times v_q)
$$

Hadamard product:

$$
p \odot q = (p^e q^e, p^i q^i, p^j q^j, p^k q^k)
$$

Inner product:

$$
p \cdot q = p^e q^e + p^i q^i + p^j q^j + p^k q^k
$$

Conjugate:

$$
p^* = (p^e, -p^i, -p^j, -p^k)
$$

Norm:

$$
|p| = \sqrt{(p^e)^2 + (p^i)^2 + (p^j)^2 + (p^k)^2}
$$

Scalar:

$$
\lambda p = (\lambda p^e, \lambda p^i, \lambda p^j, \lambda p^k)
$$

B. Construction of the QCNN

The whole structure of the QCNN is shown in Fig. 1. The
basic architecture is the same as that of the conventional
convolutional neural networks (CNNs). The QCNN has \(L\)
layers. The 0-th layer is the input-terminal layer. The last
\(L\)-th layer is the output layer. All the signals, weights and
neurons are in the quaternion domain except for the real-
valued \(L\)-th layer, which is explained later. The output values
of the network are compared with the output teacher signals
for learning, which is explained below. After the completion
of the learning, classification is conducted.

In Fig. 1, \(c\) is the channel index for the \(l\)-th layer’s input
\(X^{(l)}\) in the convolutional layer, \(k\) is the index for kernels, \((p, q)\), \((a, b)\) and \((m, n)\) are positions in the kernels, the output of
the convolutional layer and the output of the pooling layer,
respectively, for each channel.
C. Forward processing

1) Convolutional layer: The inputs and outputs of the $l$-th layer are denoted $X^{(l-1)} = [x_{a+p,b+q,c}]$ and $V^{(l)} = [v_{a,b,k}]$, respectively, and the connection weights of convolutional kernels are represented as $W^{(l)} = [w_{p,q,c,k}]$, whose height and width are denoted $P_{conv}^{(l)}$ and $Q_{conv}^{(l)}$. The quaternion convolution is defined as

$$ V^{(l)} = f(U^{(l)}) $$

where $U^{(l)} = [u_{a,b,k}]$ and $b^{(l)}$ are the neural inner state and bias, respectively. The function $f$ are an activation function defined as

$$ f(p) = f(p^i)i + f(p^j)j + f(p^k)k $$

where $p$ is a purely imaginary quaternion and $f$ is rectified linear unit (ReLU) in our experiments.

2) Pooling layer: In our experiments, the pooling processing averages the inputs as

$$ x_{m,n,k}^{(l)} = \frac{1}{P_{pool}^{(l)} \cdot Q_{pool}^{(l)}} \sum_{(a,b) \in R_{m,n}} v_{a,b,k}^{(l)} $$

where $P_{pool}^{(l)}$ and $Q_{pool}^{(l)}$ are the height and width of the pooling window $R_{m,n}$ to calculate $x_{m,n,k}^{(l)}$, respectively. After the quaternion signals pass through convolutional layers and pooling layers, they are flattened and fed into quaternion fully-connected layers, whose processing follows Ref. [18].

3) Connection between a quaternion fully-connected layer and a real-valued output layer: The output layer is a real-valued fully-connected layer directly following the last quaternion hidden layer. Before the quaternion signals are fed into the real-valued output layer, they are flattened to real-valued signals as

$$ \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \rightarrow \begin{bmatrix} x_1^r \\ x_1^i \\ x_2^r \\ x_2^i \\ \vdots \\ x_d^r \\ x_d^i \end{bmatrix} $$

where $x_d$ ($d = 1, 2, \cdots$) are purely imaginary quaternions whose imaginary components are $x_{d}^{i}$, $x_{d}^{j}$, and $x_{d}^{k}$.

Finally, the output of the QCNN goes through a softmax function, and the loss $E$ is calculated with the output teacher signal $t$, which is a one-hot vector. In our experiments, the loss $E$ is the following cross entropy:

$$ E = \sum_{j=1}^{J} -t_j \log y_j $$

where $t_j$ denotes the $j$-th component of the teacher signal $t$ and $y_j$ presents the output of the $j$-th neuron of the output layer.

D. Backward propagation in learning

1) Connection between a quaternion fully-connected layer and a real-valued output layer: Before real-valued gradients of the loss $E$ in terms of the real-valued signals are passed
the gradient given by the next layer as
\[ \frac{\partial E}{\partial \mathbf{x}_d} = \left[ \begin{array}{c} \frac{\partial E}{\partial x_1} \\
\frac{\partial E}{\partial x_2} \\
\vdots \end{array} \right] \rightarrow \left[ \begin{array}{c} \frac{\partial E}{\partial x_1} \\
\frac{\partial E}{\partial x_2} \\
\vdots \end{array} \right] \]
(19)
where \( \partial E/\partial x_d, \partial E/\partial x_d \) and \( \partial E/\partial x_d \) \( (d = 1, 2, \ldots) \) are converted to a single quaternion \( \partial E/\partial x_d \). After this conversion, the backpropagation dynamics in quaternion fully-connected layers follows Ref. [18].

2) Pooling layer: In the \( l \)-th pooling layer, \( \partial E/\partial x_{a,b,k} \) propagates from the \((l + 1)\)-th layer. There are no trainable parameters in the pooling layer. However, we have to transfer the gradient given by the next layer as
\[ \frac{\partial E}{\partial \mathbf{x}_{a,b,k}} = \frac{1}{P_{\text{pool}} L_{\text{pool}} Q_{\text{pool}}} \frac{\partial E}{\partial \mathbf{x}_{m,n,k}} \quad ((a, b) \in R_{m,n}) \]  
(20)

3) Convolutional layer: Based on the chain rule in Ref. [17], the gradient of a loss function \( E \) in terms of \( l \)-th-layer-kernel parameters is calculated as
\[ \frac{\partial E}{\partial \mathbf{w}_{p,q,c,k}} = \sum_{a,b} \frac{1}{|\mathbf{w}_{p,q,c,k}|} \cdot \frac{\partial E}{\partial \mathbf{u}_{a,b,k}} \cdot \left( \mathbf{u}_{p,q,c,k} \otimes \mathbf{x}_{a+p,b+q,c} \right) \]
\[ -2\frac{\partial E}{\partial \mathbf{u}_{a,b,k}} \otimes \mathbf{u}_{p,q,c,k} \otimes \mathbf{x}^{*(l-1)}_{a+p,b+q,c} \]
and
\[ \frac{\partial E}{\partial b_{k}} = \sum_{a,b} \frac{\partial E}{\partial \mathbf{u}_{a,b,k}} \]
(21)
where
\[ \frac{\partial E}{\partial \mathbf{u}_{a,b,k}} = f'(u_{a,b,k}) \otimes \frac{\partial E}{\partial \mathbf{u}_{a,b,k}} \]
(22)

The gradient which propagates to the previous layer is calculated as (24), where \( S_{m,n} \) is a set of points \((a, b)\) connected to \( x_{m,n,k}^{(l-1)} \).

III. EXPERIMENTS
A. The datasets and the areas where we obtained training data and test data
PolSAR images in our experiments are a 4,000 × 5,000-pixel image of Ishikari Bay and a 8,000 × 3,000-pixel image of Tomakomai, Hokkaido, Japan obtained by Japan Aerospace Exploration Agency (JAXA) Advanced Land Observing Satellite (ALOS)-2, observed in August 2015 (off nadir angle: 25.4°, ascending, level 1.1 (no averaging)). The RGB visualization of these two PolSAR images is shown in Fig. 2. We classify their pixels into four classes (water, town, grass and forest).

Fig. 3 shows the corresponding optical images obtained by Sentinel-2 in September 2017. They also present the areas where we obtained training data (unframed) and test data (framed) are also shown in Fig. 3. We place a data window with a height and a width of both 15 pixels. We pick up 600-window training data for each class (total 2,400) from the unframed areas in the Ishikari image in Fig. 3(a). The 2,400 test data (600 for each class) are obtained from the framed areas in each of the two PolSAR images so that we obtain 4,800 test data in total.

B. The features used in the experiments
The experimental features are derived from the complex scattering matrix expressed as
\[ S = \begin{bmatrix} S_{HH} & S_{HV} \\ S_{VH} & S_{VV} \end{bmatrix} \]
(25)
We obtain two sets of features, namely, the PolSAR pseudo-color features and normalized Stokes vectors, from the above scattering matrix. In our experiments, we train three types of neural networks, namely, RVCNNs, fixed-axis QCNNs and our proposed QCNNs with these two sets of features.

1) The PolSAR pseudo-color image: The scattering matrix can be decomposed under the Pauli decomposition as
\[ a = \frac{S_{HH} + S_{VV}}{\sqrt{2}}, b = \frac{S_{HH} - S_{VV}}{\sqrt{2}}, c = \frac{S_{HV} + S_{VH}}{\sqrt{2}} \]
(26)
The PolSAR pseudo-color features are constructed by assigning \(|b|^2, |c|^2\) and \(|a|^2\) to red, green and blue components, respectively [23]. They are employed in some methods as the classification features [22], [24], [25]. Intensity features such as PolSAR pseudo-color features have so fine texture of land surface that neural networks are able to extract the spatial texture by learning them. SAR images are often multi-look processed to mitigate speckle noise. However, the fine textures of a SAR image are sacrificed by the averaging processing. We do not perform the multi-look processing on the PolSAR images to extract finer textures.

When the PolSAR pseudo-color features are fed into the networks, they should be normalized to a certain range. In general, some pixels of a PolSAR image have very large amplitude values compared to others. If we try to normalize the feature values in such a manner that their maximum value becomes unity, almost all the other amplitude values become zero. Therefore, we normalize them in another way that their 90 percentile values in an image becomes unity and that, after the normalization, the amplitude value larger than unity is set as unity.

The architectures of the neural networks learning the PolSAR pseudo-color features are shown in Fig. 4. All the neural networks consist of one convolutional layer, one pooling layer, one hidden fully-connected layer and the output layer. The pooling layer conducts average pooling to mitigate speckle noise. Since in the sliding window classification, the positional relationship on the feature map is not important for classification, the pooling layer averages the entire feature map.
2) Stokes vectors normalized by their total power: Here, we explain how to derive the Stokes vectors normalized by their total power. First, we consider six incidence polarizations $E_i = \begin{bmatrix} E_{\text{HH}}^i & E_{\text{HV}}^i & E_{\text{VH}}^i & E_{\text{VV}}^i \end{bmatrix}^T$ (T: transpose), namely, horizontal polarization, vertical polarization, $+45^\circ$ linear polarization, $-45^\circ$ linear polarization, left-handed circular polarization and right-handed circular polarization in our experiments. Then, the received waves $E'' = \begin{bmatrix} E_{\text{HH}}'' & E_{\text{HV}}'' \end{bmatrix}$ are expressed as
\[
\begin{bmatrix} E_{\text{HH}}'' \\ E_{\text{HV}}'' \end{bmatrix} = \begin{bmatrix} S_{\text{HH}} & S_{\text{HV}} \\ S_{\text{VH}} & S_{\text{VV}} \end{bmatrix} \begin{bmatrix} E_{\text{HH}}^i \\ E_{\text{HV}}^i \end{bmatrix}
\] (27)

Then, the corresponding Stokes vectors $g$ are obtained as
\[
g = \begin{bmatrix} \langle g_0 \rangle \\ \langle g_1 \rangle \\ \langle g_2 \rangle \\ \langle g_3 \rangle \end{bmatrix} = \begin{bmatrix} \langle E_{\text{HH}}'' E_{\text{HH}}^* + E_{\text{HV}}'' E_{\text{HV}}^* \rangle \\ \langle E_{\text{HV}}'' E_{\text{HH}}^* - E_{\text{HH}}'' E_{\text{HV}}^* \rangle \\ \langle E_{\text{HH}}'' E_{\text{VH}}^* - E_{\text{VH}}'' E_{\text{HH}}^* \rangle \\ \langle j(E_{\text{HV}}'' E_{\text{VH}}^* - E_{\text{VH}}'' E_{\text{HV}}^*) \rangle \end{bmatrix}
\] (28)

where $\langle \cdot \rangle$ denotes spatial averaging process. Finally, we normalize the Stokes vectors by their total power $\langle g_0 \rangle$ as
\[
P = \begin{bmatrix} \langle g_1 \rangle / \langle g_0 \rangle \\ \langle g_2 \rangle / \langle g_0 \rangle \\ \langle g_3 \rangle / \langle g_0 \rangle \end{bmatrix}
\] (29)

The normalized Stokes vectors have the Degree of Polarization (DoP) information in their norm as
\[
\text{DoP} = \sqrt{\frac{\langle g_1 \rangle^2 + \langle g_2 \rangle^2 + \langle g_3 \rangle^2}{\langle g_0 \rangle^2}}
\] (30)

In general, scattered waves are partially polarized, and it is very important to learn the DoP information, which is totally different from intensity.

The architectures of the neural networks learning the normalized Stokes vectors are shown in Fig. 5. We determined them empirically.

C. Learning and classification processing

The neural networks are trained with a minibatch having 40 data. One epoch has 2,400/40 = 60 times weight updates. We train the neural networks learning the PolSAR pseudo-color features for 1,000 epochs and those learning the normalized Stokes vectors for 200 epochs. The loss function is calculated as the cross-entropy loss in (18). The weight updating is conducted by stochastic gradient descent with momentum whose learning rate is 0.01 and momentum parameter is 0.9 in our experiments.

In classification processing, a 15 $\times$ 15-pixel-window image cropped from the PolSAR image is fed into the neural networks one-by-one. Then, the loss is calculated by the outputs and the output teacher signals for each class, and the class having minimum loss is the class which the pixel belongs to. This process is conducted until all the pixels are classified.
D. Combining classification results obtained for the two sets of features

We explain a method of combining two classification results obtained for the two different sets of features here. First, a neural network trained with each set of features classifies both of the PolSAR images. In the classification processing, we obtain the losses in (18) per pixel. Then, the two losses based on two different features are compared pixel-to-pixel, and the class with the smaller loss is regarded as the final result. In other words, the class of the merged result $c_m$ is obtained as

$$c_m = \begin{cases} c_1 & (E_1 < E_2) \\ c_2 & (E_1 > E_2) \end{cases}$$

Fig. 3: (a) Ishikari Bay and (b) Tomakomai observed by Sentinel-2 optical images including the areas from which we obtained training data (unframed) and test data (framed.)

Fig. 4: The architectures of the three neural networks learning the PolSAR pseudo-color, namely, (a) RVCNN, (b) fixed-axis QCNN and (c) QCNN. The symbols in the left side of this figure correspond to parameters in Fig. 1.

Fig. 5: The architectures of the three neural networks learning the Stokes vectors normalized by their total power, namely, (a) RVCNN, (b) fixed-axis QCNN and (c) QCNN. The symbols in the left side of this figure correspond to parameters in Fig. 1.
where $c_i$ and $E_i$ ($i = 1, 2$) are the classification result and loss based on feature $i$, respectively.

IV. RESULTS

A. The classification results for the PolSAR pseudo-color features

Fig. 6 shows the classification results of the PolSAR pseudo-color features obtained for Ishikari scene. It is found that the sea area in the upper left is significantly misclassified into grass. Since the backscattering in the sea and grass is very small, the neural networks learning intensity features such as PolSAR pseudo-color features cannot differentiate between them. In addition, there are some misclassifications between forest and town in the lower left of the results. In general, town and forest are misclassified each other because of their similar scattering characteristics.

Fig. 7 shows the classification results of Tomakomai by training data obtained from the Ishikari image. The town area in the upper right is misclassified into forest while the forest area in the lower left into town. The buildings in this town area have such a large angle to the direction of the radar beam that the cross-polarization components of scattering in this town area, namely, $S_{HV}$ and $S_{VH}$, become relatively large like forest. Therefore, it is very difficult to discriminate between town and forest generally. Fig. 8 shows the enlarged views of this town area. Our proposed QCNN classifies it more accurately than the RVCNN and fixed-axis QCNN because it can learn the relationship between the components of input vectors by arbitrary quaternion rotations.

Table I presents the confusion matrices and F-scores of respective neural networks measured by test data obtained from both of the two PolSAR images. From the confusion matrices, it can be seen that the water and grass, and the town and forest are misclassified each other, respectively. Note that, for the PolSAR pseudo-color features, we find much misclassification between the water and the grass. The F-scores show that our proposed QCNN especially succeeded in discriminating between town and forest compared to the other two networks. Our proposed QCNN shows the best average F-score.

B. The classification results for the normalized Stokes vectors

Fig. 9 shows the classification results of Ishikari for the normalized Stokes vectors. The sea area in the upper left of the classification results is accurately classified compared to that of the results for the PolSAR pseudo-color features. Scattering waves at water surface have low coherence due to ripples while possible wind-swaying grass is invisible by an L-band electromagnetic wave. Therefore, the DoP values for these two areas become different. The neural networks are able to differentiate between the sea and grass by learning partially polarized state from the normalized Stokes vectors.

Fig. 10 shows the classification results of Tomakomai for the normalized Stokes vectors. The town area in the upper right is significantly misclassified into forest. The town and forest have so similar scattering characteristics in pixels that they need to be discriminated by spatial texture. The fine textures of PolSAR images are included mainly in intensity features. However, the Stokes vectors are normalized by their intensity, so that the textures are lost, coupled with the impact of multi-look processing. Therefore, the neural networks cannot learn spatial textures, and fail to discriminate between town and forest.

Table II presents the confusion matrices and F-scores of respective neural networks. Compared to Table I for the PolSAR pseudo-color features, all the neural networks succeeded in discriminating between water and grass while they fail to distinguish between town and forest. From these two experiments for the two different features, it is found that the classification results largely depend on learned features.

C. Our proposed merging results

The previous results show that the PolSAR pseudo-color features are suitable for discrimination between town and forest, but not between sea and grass, while the normalized Stokes vectors are opposite. Fig. 11 shows the merged results of Ishikari using the two features. The sea area seems to be accurately classified. Fig. 12 presents the classification results of sea area in Ishikari shown so far for respective neural networks and features. Compared to the classification results for the PolSAR pseudo-color features, the sea area quality of merged results is largely improved. In addition, in Fig. 13, the forest area in the lower left is more accurately classified compared to the classification results for each feature.

The merging results of Tomakomai area are shown in Fig. 14. The forest area in the lower left and the town area in the upper right are accurately classified. Fig. 15 shows the classification results of the forest area in Tomakomai presented so far for respective neural networks and features. The forest area quality of the merged results is improved compared to the result for each feature respectively. Fig. 16 summarizes the classification results of the town area in the upper right in Tomakomai. The merged results of this town area are worse than the results for the PolSAR pseudo-color features while they are better than the those for the normalized Stokes vectors.

Table III presents the confusion matrices and F-scores of respective neural networks. Compared to Tables I and II, the overall classification results are improved by the merging method. Table IV summarizes the average F-scores of respective neural networks and features presented so far. The results of our proposed QCNN shows the best classification performance in all the situations. In addition, the merging method significantly improved the classification results, demonstrating its effectiveness.

V. CONCLUSION

We proposed full-learning quaternion convolutional neural networks which learn data by updating all four parameters of a quaternion. The proposed QCNNs showed better classification performance than the conventional RVCNNs and fixed-axis QCNNs. This is because the full-learning QCNN can learn the relationship between the components of input vectors with arbitrary quaternion rotations. In addition, we showed
TABLE I: The classification results of RVCNN, fixed-axis QCNN and QCNN for PolSAR pseudo-color features.

<table>
<thead>
<tr>
<th>Neural network</th>
<th>Class</th>
<th>water</th>
<th>town</th>
<th>grass</th>
<th>forest</th>
<th>F-score</th>
<th>avg F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVCNN</td>
<td>water</td>
<td>1009</td>
<td>0</td>
<td>191</td>
<td>0</td>
<td>0.846</td>
<td>0.866</td>
</tr>
<tr>
<td></td>
<td>town</td>
<td>0</td>
<td>1078</td>
<td>5</td>
<td>117</td>
<td>0.939</td>
<td></td>
</tr>
<tr>
<td></td>
<td>grass</td>
<td>177</td>
<td>1</td>
<td>890</td>
<td>132</td>
<td>0.779</td>
<td></td>
</tr>
<tr>
<td></td>
<td>forest</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>1184</td>
<td>0.899</td>
<td></td>
</tr>
<tr>
<td>fixed-axis QCNN</td>
<td>water</td>
<td>943</td>
<td>0</td>
<td>257</td>
<td>0</td>
<td>0.826</td>
<td>0.845</td>
</tr>
<tr>
<td></td>
<td>town</td>
<td>0</td>
<td>1020</td>
<td>12</td>
<td>168</td>
<td>0.903</td>
<td></td>
</tr>
<tr>
<td></td>
<td>grass</td>
<td>139</td>
<td>3</td>
<td>931</td>
<td>127</td>
<td>0.775</td>
<td></td>
</tr>
<tr>
<td></td>
<td>forest</td>
<td>0</td>
<td>35</td>
<td>3</td>
<td>1162</td>
<td>0.875</td>
<td></td>
</tr>
<tr>
<td>QCNN</td>
<td>water</td>
<td>994</td>
<td>0</td>
<td>206</td>
<td>0</td>
<td>0.849</td>
<td>0.874</td>
</tr>
<tr>
<td></td>
<td>town</td>
<td>0</td>
<td>1117</td>
<td>9</td>
<td>74</td>
<td>0.952</td>
<td></td>
</tr>
<tr>
<td></td>
<td>grass</td>
<td>148</td>
<td>1</td>
<td>919</td>
<td>132</td>
<td>0.787</td>
<td></td>
</tr>
<tr>
<td></td>
<td>forest</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>1172</td>
<td>0.909</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II: The classification results of RVCNN, fixed-axis QCNN and QCNN for normalized Stokes vectors.

<table>
<thead>
<tr>
<th>Neural network</th>
<th>Class</th>
<th>water</th>
<th>town</th>
<th>grass</th>
<th>forest</th>
<th>F-score</th>
<th>avg F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVCNN</td>
<td>water</td>
<td>1192</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>0.975</td>
<td>0.884</td>
</tr>
<tr>
<td></td>
<td>town</td>
<td>0</td>
<td>942</td>
<td>46</td>
<td>212</td>
<td>0.804</td>
<td></td>
</tr>
<tr>
<td></td>
<td>grass</td>
<td>53</td>
<td>177</td>
<td>939</td>
<td>31</td>
<td>0.856</td>
<td></td>
</tr>
<tr>
<td></td>
<td>forest</td>
<td>0</td>
<td>21</td>
<td>2</td>
<td>1177</td>
<td>0.898</td>
<td></td>
</tr>
<tr>
<td>fixed-axis QCNN</td>
<td>water</td>
<td>1184</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>0.977</td>
<td>0.881</td>
</tr>
<tr>
<td></td>
<td>town</td>
<td>0</td>
<td>979</td>
<td>49</td>
<td>172</td>
<td>0.808</td>
<td></td>
</tr>
<tr>
<td></td>
<td>grass</td>
<td>39</td>
<td>210</td>
<td>905</td>
<td>46</td>
<td>0.835</td>
<td></td>
</tr>
<tr>
<td></td>
<td>forest</td>
<td>0</td>
<td>27</td>
<td>4</td>
<td>1169</td>
<td>0.904</td>
<td></td>
</tr>
<tr>
<td>QCNN</td>
<td>water</td>
<td>1185</td>
<td>10</td>
<td>5</td>
<td>0</td>
<td>0.964</td>
<td>0.890</td>
</tr>
<tr>
<td></td>
<td>town</td>
<td>0</td>
<td>955</td>
<td>66</td>
<td>179</td>
<td>0.821</td>
<td></td>
</tr>
<tr>
<td></td>
<td>grass</td>
<td>74</td>
<td>149</td>
<td>965</td>
<td>12</td>
<td>0.859</td>
<td></td>
</tr>
<tr>
<td></td>
<td>forest</td>
<td>0</td>
<td>13</td>
<td>12</td>
<td>1175</td>
<td>0.916</td>
<td></td>
</tr>
</tbody>
</table>

TABLE III: The merged results of RVCNN, fixed-axis QCNN and QCNN.

<table>
<thead>
<tr>
<th>Neural network</th>
<th>Class</th>
<th>water</th>
<th>town</th>
<th>grass</th>
<th>forest</th>
<th>F-score</th>
<th>avg F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVCNN</td>
<td>water</td>
<td>1196</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0.970</td>
<td>0.921</td>
</tr>
<tr>
<td></td>
<td>town</td>
<td>0</td>
<td>1052</td>
<td>9</td>
<td>139</td>
<td>0.925</td>
<td></td>
</tr>
<tr>
<td></td>
<td>grass</td>
<td>69</td>
<td>11</td>
<td>982</td>
<td>138</td>
<td>0.895</td>
<td></td>
</tr>
<tr>
<td></td>
<td>forest</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>1189</td>
<td>0.892</td>
<td></td>
</tr>
<tr>
<td>fixed-axis QCNN</td>
<td>water</td>
<td>1176</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>0.964</td>
<td>0.916</td>
</tr>
<tr>
<td></td>
<td>town</td>
<td>0</td>
<td>1044</td>
<td>8</td>
<td>148</td>
<td>0.919</td>
<td></td>
</tr>
<tr>
<td></td>
<td>grass</td>
<td>63</td>
<td>17</td>
<td>988</td>
<td>132</td>
<td>0.889</td>
<td></td>
</tr>
<tr>
<td></td>
<td>forest</td>
<td>0</td>
<td>10</td>
<td>2</td>
<td>1188</td>
<td>0.891</td>
<td></td>
</tr>
<tr>
<td>QCNN</td>
<td>water</td>
<td>1196</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0.970</td>
<td>0.928</td>
</tr>
<tr>
<td></td>
<td>town</td>
<td>0</td>
<td>1096</td>
<td>14</td>
<td>90</td>
<td>0.946</td>
<td></td>
</tr>
<tr>
<td></td>
<td>grass</td>
<td>69</td>
<td>13</td>
<td>980</td>
<td>138</td>
<td>0.890</td>
<td></td>
</tr>
<tr>
<td></td>
<td>forest</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>1187</td>
<td>0.908</td>
<td></td>
</tr>
</tbody>
</table>

TABLE IV: The summary of the average F-scores for each neural network and each feature.

<table>
<thead>
<tr>
<th>Feature</th>
<th>RVCNN</th>
<th>fixed-axis QCNN</th>
<th>QCNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PolSAR pseudo-color</td>
<td>0.865</td>
<td>0.845</td>
<td>0.874</td>
</tr>
<tr>
<td>Normalized Stokes vectors</td>
<td>0.883</td>
<td>0.881</td>
<td>0.890</td>
</tr>
<tr>
<td>Merge</td>
<td>0.921</td>
<td>0.916</td>
<td>0.928</td>
</tr>
</tbody>
</table>

that the classification results largely depend on input features. Specifically, the PolSAR pseudo-color features are suitable for discrimination between town and forest, but not between water and grass, while the normalized Stokes vectors are opposite. We merged the classification results for respective features based on their cross-entropy values to highly improve the
final classification by compensating for the shortcomings of respective features successfully.

REFERENCES


Fig. 6: Classification results of Ishikari scene for PolSAR pseudo-color features by (a)RVCNN, (b)fixed-axis QCNN and (c)QCNN as well as (d)Sentinel-2 optical image for reference.
Fig. 7: Classification results of Ishikari scene for PolSAR pseudo-color features by (a)RVCNN, (b)fixed-axis QCNN and (c)QCNN as well as (d)Sentinel-2 optical image for reference.
Fig. 8: Classification results of Ishikari town area for PolSAR pseudo-color features by (a)RVCNN, (b)fixed-axis QCNN and (c)QCNN as well as (d)Sentinel-2 optical image for reference.
Fig. 9: Classification results of Ishikari scene for normalized Stokes vectors by (a)RVCNN, (b)fixed-axis QCNN and (c)QCNN as well as (d)Sentinel-2 optical image.
Fig. 10: Classification results of Tomakomai scene for normalized Stokes vectors by (a) RVCNN, (b) fixed-axis QCNN and (c) QCNN as well as (d) Sentinel-2 optical image.
Fig. 11: Merged classification results of Ishikari scene by (a)RVCNN, (b)fixed-axis QCNN and (c)QCNN as well as (d)Sentinel-2 optical image.
Fig. 12: The classification results of sea area in Ishikari by respective network and features as well as the corresponding area of a Sentinel-2 optical image.

Fig. 13: The classification results of forest area in Ishikari by respective networks and features as well as the corresponding area of a Sentinel-2 optical image.
Fig. 14: Merged classification results of Tomakomai scene of (a)RVCNN, (b)fixed-axis QCNN and (c)QCNN as well as (d)Sentinel-2 optical image.
Fig. 15: The classification results of forest area in Tomakomai by respective networks and features as well as the corresponding area of a Sentinel-2 optical image.

Fig. 16: The classification results of town area in Tomakomai by respective networks and features as well as the corresponding area of a Sentinel-2 optical image.