Research Article

Compressed Wavelet Tensor Attention Capsule Network

Xiushan Liu, Chun Shan, Qin Zhang, Jun Cheng, and Peng Xu

School of Electronics and Information Engineering, Guangdong Polytechnic Normal University, Guangzhou 510665, Guangdong, China

Correspondence should be addressed to Chun Shan; shanchun@gpnu.edu.cn

Received 11 March 2021; Revised 22 March 2021; Accepted 3 April 2021; Published 17 April 2021

Academic Editor: Muhammad Shafiq

Copyright © 2021 Xiushan Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Texture classification plays an important role for various computer vision tasks. Depending upon the powerful feature extraction capability, convolutional neural network (CNN)-based texture classification methods have attracted extensive attention. However, there still exist many challenges, such as the extraction of multilevel texture features and the exploration of multidirectional relationships. To address the problem, this paper proposes the compressed wavelet tensor attention capsule network (CWTACapsNet), which integrates multiscale wavelet decomposition, tensor attention blocks, and quantization techniques into the framework of capsule neural network. Specifically, the multilevel wavelet decomposition is in charge of extracting multiscale spectral features in frequency domain; in addition, the tensor attention blocks explore the multidimensional dependencies of convolutional feature channels, and the quantization techniques make the computational storage complexities be suitable for edge computing requirements. The proposed CWTACapsNet provides an efficient way to explore spatial domain features, frequency domain features, and their dependencies which are useful for most texture classification tasks. Furthermore, CWTACapsNet benefits from quantization techniques and is suitable for edge computing applications. Experimental results on several texture datasets show that the proposed CWTACapsNet outperforms the state-of-the-art texture classification methods not only in accuracy but also in robustness.

1. Introduction

Texture classification is crucial in pattern recognition and computer vision [1–5]. Since many very sophisticated classifiers exist, the key challenge here is the development of effective features to extract from a given textured image [6]. As an important research issue, many methods have been proposed to represent texture features [7, 8]. About 51 different sets of texture features are summarized in [9]. These texture features are generally hand-crafted under some hypothesis of texture characteristics. Because different texture datasets contain different types of textures, the performance of hand-crafted features is usually changed for different datasets [4].

Recently, texture representation methods based on CNN have been achieved powerful representation capability [6, 10–13]. These CNN-based methods implement texture feature extraction in an end-to-end way which does not require predefined representation formula. Moreover, Fujieda et al. [11] find that integrating wavelet transform into CNN can effectively capture spectral information of texture images. Nevertheless, there still exist many challenges, such as extracting multilevel texture features and capturing sufficient relationships [11]. Pooling operations of CNN-based methods proactively discard substantial information which prevents the efficient exploration for texture feature relationships [14, 15]. In contrast, the capsule neural network (CapsNets), which implements dynamic routing algorithm instead of traditional pooling mechanism, can probably get rid of the weakness of pooling operations. In addition, CapsNets replaces scalar outputs of CNN with more informative vector outputs obtained by the squashing activation function. CapsNets has several advantages, such as relationship awareness and stable generalization capability [16, 17].

The attention mechanism [18] is proposed to help models to focus on more relevant regions, capture complex correlations, and discover new patterns within images.
Therefore, the integration of attention mechanism and CapsNets has great potential to represent texture features and explore their relationships sufficiently. The key problem preventing attention mechanism and CapsNets from being applied in edge computing domain is that they both suffer from heavily computation and memory burdens. It is essential to consider the quantization techniques for deploying models on edge devices [19–33]. This paper proposes the compressed wavelet tensor attention capsule network (CWTACapsNet) that integrates multilevel wavelet decomposition, tensor attention mechanism, and quantization techniques into the capsule network.

The proposed CWTACapsNet involves several compressed multiscale tensor self-attention blocks that can capture multidirectional dependencies across different channels. Furthermore, CWTACapsNet utilizes Nyström technique and proposes quantized dynamic routing process to release resource requirements. The main contributions of CWTACapsNet are three folds. First, it uses multilevel wavelet transform to extract multiscale spectral features in frequency domain which further extends texture representation capability. Second, it employs tensor attention mechanism via matricization to explore the multidirectional dependencies of texture features in different scales. Third, it employs quantization techniques to reduce the computation and memory costs without sacrificing the accuracy.

The rest of the paper is organized as follows. Section 2 presents the whole architecture and key parts of the proposed CWTACapsNet. Section 3 presents validation experiments and discusses the experimental results. The conclusion is drawn in Section 4.

2. Compressed Wavelet Tensor Attention Capsule Network

The proposed CWTACapsNet integrates multiscale wavelet decomposition and tensor self-attention blocks into capsule network. The architecture of CWTACapsNet is shown in Figure 1. CWTACapsNet involves the wavelet feature extraction block, the compressed multiscale tensor self-attention block, and the quantized capsule network. The wavelet feature extraction block extracts multiscale spectral features with multilevel wavelet decomposition. The compressed tensor self-attention block captures the multidirectional relationships within each scale, and the primary capsules are generated based on the wavelet and tensor attentive information.

2.1. Multiscale Feature Extraction via Wavelet Decomposition.

Given an image \( x \), we utilize the 2D discrete wavelet transform (DWT) [34] with four convolutional filters, i.e., low-pass filter, \( f_{LL} \), and high-pass filters, \( f_{LH}, f_{HL} \), and \( f_{HH} \), to decompose \( x \) into four subband images, i.e., \( x_{LL}, x_{LH}, x_{HL}, \) and \( x_{HH} \). The convolutional stride is 2. The four filters are defined by

\[
\begin{align*}
    f_{LL} &= \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \\
    f_{LH} &= \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}, \\
    f_{HL} &= \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, \\
    f_{HH} &= \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.
\end{align*}
\]

The four filters (see equation (1)) indicate that they are orthogonal to each other and form a \( 4 \times 4 \) invertible matrix.

The DWT operation is given by

\[
\begin{align*}
    x_{LL} &= (f_{LL} \ast x)_{\downarrow 2}, \\
    x_{LH} &= (f_{LH} \ast x)_{\downarrow 2}, \\
    x_{HL} &= (f_{HL} \ast x)_{\downarrow 2}, \\
    x_{HH} &= (f_{HH} \ast x)_{\downarrow 2},
\end{align*}
\]

where * denotes convolution operator and \( \downarrow 2 \) denotes the downsampling with stride 2. The \( (i, j) \)-th value of \( x_{LL}, x_{LH}, x_{HL}, \) and \( x_{HH} \) after 2D Haar transform [19] is given by

\[
\begin{align*}
    x_{LL}(i, j) &= x(2i - 1, 2j - 1) + x(2i - 1, 2j) + x(2i, 2j - 1) + x(2i, 2j), \\
    x_{LH}(i, j) &= -x(2i - 1, 2j - 1) - x(2i - 1, 2j) + x(2i, 2j - 1) + x(2i, 2j), \\
    x_{HL}(i, j) &= -x(2i - 1, 2j - 1) + x(2i - 1, 2j) - x(2i, 2j - 1) + x(2i, 2j), \\
    x_{HH}(i, j) &= x(2i - 1, 2j - 1) - x(2i - 1, 2j) - x(2i, 2j - 1) + x(2i, 2j).
\end{align*}
\]

Based on multilevel wavelet package transform [35], the subband image \( x_{LL} \) is recursively decomposed by DWT. Because the downsampling stride is 2, the sizes of extracted subband images in different wavelet decomposition levels are diminished gradually. In addition, the upsampling operations (with stride (2) are employed to guarantee the size consistency of convolution feature maps for tensor concatenation.

2.2. Compressed Tensor Self-Attention Block.

Inspired by [36], we design the compressed tensor self-attention block based on matricization and Nyström technique. The matricization can capture interdependencies along all dimensions of tensorized convolution feature maps. To reduce the computational and storage requirement of attention computation, we use Nyström technique to achieve an
approach, an approximation solution which releases the resource burden of inference and speeds up significantly.

These tensorized convolution feature maps are generated based on wavelet-extracted features. The input 3rd-order tensor can be viewed as a combination of its three mode-matricizations. Combining their outputs allows the compressed tensor self-attention block to make use of interchannel and intrachannel interdependencies. Moreover, the Nyström-based self-attention module involved in the compressed tensor self-attention block implements the self-attention computation to explore dependencies along corresponding mode in a more efficient way. The architectures of the compressed tensor self-attention block and the Nyström-based self-attention module are shown in Figure 2.

A mode-$n$-matricization of 3rd-order input tensor $F_i$, $i \in \{0, 1, 2, 3\}$, is the vector obtained by fixing all indices of $F_i$ except for the $n$th dimension and can be seen as a generalization of matrix's rows and columns, $n \in \{1, 2, 3\}$. The mode-$n$-matricization of 3rd-order tensor $F_i \times l_1 \times l_2$ is a case of matricization denoted as $X_{(n)}$ and arranges its mode-$n$-fibers to be the columns of the resulting matrix.

To simplify notations, we ignore the subscript. Let $X \in \mathbb{R}^{l_{1}l_{2}l_{3}}$ be the input matrix of the self-attention module, and it is projected using three matrices $W_V \in \mathbb{R}^{l_{1}l_{2}l_{3}}, W_K \in \mathbb{R}^{l_{1}l_{2}l_{3}},$, and $W_Q \in \mathbb{R}^{l_{1}l_{2}l_{3}}$ to extract feature representations $Q \in \mathbb{R}^{h_{q}l_{3}}$, $K \in \mathbb{R}^{h_{k}l_{3}}$, and $V \in \mathbb{R}^{h_{v}l_{3}}$ as follows:

$$Q = XW_Q,$$
$$K = XW_K,$$
$$V = XW_V.\tag{4}$$

The output of the self-attention module is computed by

$$O = V + \alpha \text{softmax} \left( \frac{QK^T}{\sqrt{m}} \right)V, \tag{5}$$

where $\alpha > 0$ denotes the learnable coefficient and softmax(-) denotes a row-wise softmax normalization function. Then, generate tensor $Y$ by reshaping $O$ as tensor form.

As shown in equation (5), the self-attention mechanism requires calculating $h^2$ similarity scores between each pair of vectors, resulting in a complexity of $O(h^2)$ for both memory and time. Due to this quadratic dependence on the input length, the application of self-attention is limited to small size matrices (e.g., $h < 1000$) for edge devices. It is necessary to reduce the resource burden. Inspired by [37], we utilize Nyström technique to build a resource-efficient self-attention module. We rewrite the softmax operation in equation (5) as follows:

$$S = \text{softmax} \left( \frac{QK^T}{\sqrt{m}} \right) = \begin{bmatrix} A_S & B_S \\ F_S & C_S \end{bmatrix}_{l_{3}l_{3}h_{q}h_{k}}. \tag{6}$$
where \( A_S \in \mathbb{R}^{m \times m}, B_S \in \mathbb{R}^{m \times (h - m)}, F_S \in \mathbb{R}^{(h - m) \times m}, \) and \( C_S \in \mathbb{R}^{(h - m) \times (h - m)}. \) \( A_S \) denotes the selected matrix generated by sampling \( m \) columns and \( m \) rows from matrix \( S \) via some adaptive sampling strategy [38].

According to the Nyström method [37, 38], \( S \) can be approximated by

\[
\tilde{S} = \begin{bmatrix} A_S & B_S \\ F_S & C_S \end{bmatrix} = \begin{bmatrix} A_S & B_S \\ F_S & F_S A_S^T B_S \end{bmatrix}_{h \times h} \begin{bmatrix} A_S & B_S \end{bmatrix}_{h \times h},
\]

where \( A_S \) denotes the pseudoinverse (Moore–Penrose inverse) of \( A_S \), and \( C_S \) is approximated by \( F_S A_S^T B_S \).

The SVD of \( A_S \) can be written as \( A_S = U_A \Sigma_A V_A^T \), where \( U_A V_A^T = I \) and \( \Sigma_A \) are orthogonal unitary matrices, and \( \Sigma_A \in \mathbb{R}^{m \times m} \) denotes the diagonal matrix whose diagonal elements are corresponding singular values of \( A_S \). Then, pseudoinverse \( A_S^\dagger \) can be computed by

\[
A_S^\dagger = V_A \Sigma_A^{-1} U_A^T.
\]

Submitting equations (8) into (7), we obtain

\[
\tilde{S} = \begin{bmatrix} A_S \\ F_S \end{bmatrix} V_A \Sigma_A^{-1} U_A^T \begin{bmatrix} A_S \\ B_S \end{bmatrix}.
\]

From equations (6)–(9), we can find that \( \tilde{S} \) requires all entries in \( QK^T \) due to the softmax function, even though the approximation only needs to access a subset of the columns and rows of \( S \), e.g., \( \begin{bmatrix} A_S \\ F_S \end{bmatrix} \in \mathbb{R}^{m \times m} \) corresponds to the first \( m \) columns of \( S \) (see equation (6)) and \( \begin{bmatrix} A_S \\ B_S \end{bmatrix} \in \mathbb{R}^{m \times h} \) corresponds to the first \( m \) rows of \( S \). An efficient way is to approximate \( \tilde{S} \) using subsampled matrices instead of whole one (i.e., the \( h \times h \) matrix \( QK^T \)). Let \( \tilde{K} \in \mathbb{R}^{m \times m} \) denote the matrix that consists of \( m \) columns of \( K \) and \( \tilde{Q} \in \mathbb{R}^{m \times h} \) denote the matrix that consists of \( m \) rows of \( Q \). Then, we compute the approximations as follows:
\[
\left[\begin{array}{c}
A_S \\
F_S
\end{array}\right] \approx \text{softmax}\left(\frac{QK^T}{\sqrt{m}}\right),
\]
\[
\left[\begin{array}{c}
A_S \\
B_S
\end{array}\right] \approx \text{softmax}\left(\frac{\bar{Q}K^T}{\sqrt{m}}\right),
\]
\[
A_S^i = \left(\text{softmax}\left(\frac{\bar{Q}K^T}{\sqrt{m}}\right)\right).
\]

Based on equations (7)–(9), we can obtain the efficiently approximated \( \hat{S} \) as follows:
\[
\hat{S} \approx \text{softmax}\left(\frac{QK^T}{\sqrt{m}}\right) \text{softmax}\left(\frac{\bar{Q}K^T}{\sqrt{m}}\right) \text{softmax}\left(\frac{QK^T}{\sqrt{m}}\right),
\]
where \( \bar{Q} \) and \( \bar{K}^T \) are selected before the softmax operation, which means \( \hat{S} \) can be computed only using small submatrices instead of the whole one (the \( h \times h \) matrix \( QK^T \)).

The output of each single compressed tensor self-attention module is computed by
\[
O = V + a\hat{S}V.
\]

Then, the output of the compressed tensor self-attention block (Figure 2(a)) can be generated by
\[
Z = \Psi_i\left(O^{(1)}\right)\Psi_j\left(O^{(2)}\right)\Psi_j\left(O^{(3)}\right),
\]
where \( \Psi_n \) denotes a reshape function which rearranges the matrix \( O^{(n)} \) as the tensor of dimension \( C \times H \times W \), \( n \in \{1, 2, 3\} \), and \( \Psi \) denotes the matrix concatenation operator.

2.3. Quantized Capsule Network. Aiming to overcome the deficiency and shortcoming of convolutional neural networks, a novel architecture of neural network called capsule networks was first introduced by Geoffrey Hinton [14]. A capsule is a set of neurons represented as a vector. The individual values are to capture features of an object, while the length of the vector shows the capsule activation probability. The first layer of capsules comes from the output of a convolution. This output is rearranged into vectors with a previously specified dimension (and is shrunk using the squashing function), which are used to compute the output of a next layer set of capsules. The algorithm with which the next layer capsules are computed using the current layer of capsules outputs is called dynamic routing. It takes predictions from the current layer capsules about the output of the next layer capsules and computes the actual output according to an agreement metric between predictions.

It should note that the superiority of capsule network leads to the heavy burden of computation and storage. To address this problem and make it easy to deploy on edge computing devices, we integrate share structure and quantization technique into capsule network and propose the quantized capsule network.

As shown in Figure 1, the input of quantized capsule network is generated by concatenating the output tensors of multiple compressed tensor self-attention blocks through some upsampling operation. From these concatenated tensors, the quantized convolutional layer extracts basic features. The primary capsule layer explores more detailed patterns from the extracted basic features:
\[
u_i = \text{Reshape}(QConv(Z)), \quad i = 1, \ldots, M_p,
\]
where \( u_i \in R^p \) denotes the output of capsule \( i \) in the primary capsule layer, \( p \) denotes the dimension of primary capsule vector (or capsule vector length), \( M_p \) denotes the number of capsules in the primary capsule layer, \( \text{Reshape}(\cdot) \) denotes the function that reshapes the output tensors into capsule vectors (the detailed description is provided in [14]), and \( QConv(\cdot) \) denotes the quantized convolution operator (the detailed derivation of quantized convolution is provided in [39]).

Generally, the prediction vector generated by the primary layer capsule \( i \), \( \tilde{u}_{ji} \in R^c \), indicates how much the primary layer capsule \( i \) contributes to the class layer capsule \( j \). \( \tilde{u}_{ji} \) is given by
\[
\tilde{u}_{ji} = \mathcal{W}_{ji}u_i, \quad i = 1, \ldots, M_p, \quad j = 1, \ldots, M_C,
\]
where \( \mathcal{W}_{ji} \in R^{c \times p} \) denotes the weight matrix between the primary layer capsule \( i \) and the class layer capsule \( j \), \( c \) denotes the dimension of the class layer capsule vector, \( p \) denotes the dimension of primary layer capsule vector (or capsule vector length), and \( M_C \) and \( M_p \) denote the numbers of capsules in the class layer and the primary layer capsule, respectively.

From equation (15), we can find that there are \( M_C \times M_p \) weight matrices \( \mathcal{W}_{ji} \), which leads to heavy computation and memory burden. To reduce the burden, we adopt two strategies. First, we utilize the shared structure of weight matrices (shown in Figure 3) as
\[
\tilde{u}_{ji} = \mathcal{W}_{j}u_i, \quad i = 1, \ldots, M_p, \quad j = 1, \ldots, M_C,
\]
where \( \mathcal{W}_j \in R^{c \times p} \) denotes the transformation weight matrix corresponding to class layer capsule \( j \) (i.e., each class layer capsule shares its weight matrix to all primary layer capsules). Equation (16) indicates that the number of weight matrices is reduced from \( M_C \times M_p \) to \( M_C \).

Second, we propose the quantized dynamic routing process that implements the dynamic routing in a more efficient way (shown in Figure 4). For simplicity, we assume that \( p \) can be divided by \( P (< p) \) with no reminder and \( \bar{p} = p/P \). Let \( \mathcal{W}_{j} = [\bar{w}^{(1)}, \bar{w}^{(2)}, \ldots, \bar{w}^{(P)}] \in R^{c \times \bar{p}} \) where \( \bar{w}^{(t)} \in R^{c \times \bar{p}} \) denotes the \( t \)-th submatrix of \( \mathcal{W}_{j} \), \( t = 1, \ldots, P, \quad j = 1, \ldots, M_C \). We train subcodebook for subspaces of weight matrices as follows:
Figure 3: The shared structure of weight matrices of the capsule network.

Figure 4: The workflow of quantized dynamic routing.
\[
\min_{B^{(r)}_j} \sum_{j=1}^{M_C} \left\| B^{(r)}_j (\mathcal{D}^{(r)}_j - \mathbf{w}^{(r)}_j) \right\|_F^2 \\
\text{s.t. } B^{(r)}_j \in [0,1]^K, \mathcal{D}^{(r)}_j \in R^{K \times p}, r = 1, \ldots, P,
\]

where \( \mathcal{D}^{(r)}_j \) denotes the subcodebook consists of \( K \) subcodewords for \( \mathbf{w}^{(r)}_j \), \( j = 1, \ldots, M_C \), and \( B^{(r)}_j \) denotes the indexing matrix, and each row of \( B^{(r)}_j \) only has one nonzero entry which specifies the quantization relationship between subvector and subcodebook. The alternative optimization algorithm, such as k-means clustering, is employed for learning \( \mathcal{D}^{(r)}_j \) and \( B^{(r)}_j \).

Let \( \mathbf{u}^{(1)}_\tau = \begin{bmatrix} (\mathbf{u}^{(1)}_1)^T, (\mathbf{u}^{(2)}_1)^T, \ldots, (\mathbf{u}^{(p)}_1)^T \end{bmatrix} \) where \( \mathbf{u}^{(1)}_\tau \in \mathbb{R}^p \) denotes the \( \tau \)-th subvector of \( \mathbf{u}_\tau \), \( \tau = 1, \ldots, P, i = 1, \ldots, M_P \). We train subcodebook for subspaces of primary layer capsule vectors as follows:

\[
\tilde{\mathbf{u}}_{\tau i} = \mathcal{F} \mathbf{u}_\tau = \begin{bmatrix} \tilde{\mathbf{w}}^{(1)}_\tau, \tilde{\mathbf{w}}^{(2)}_\tau, \ldots, \tilde{\mathbf{w}}^{(p)}_\tau \end{bmatrix} = \begin{bmatrix} \mathbf{u}^{(1)}_\tau \ 
\mathbf{u}^{(2)}_\tau \ 
\cdots \ 
\mathbf{u}^{(p)}_\tau \end{bmatrix} = \sum_{i=1}^{M_P} \mathbf{B}^{(r)}_i \mathcal{D}^{(r)}_i \mathcal{F}^{(r)} \mathbf{v}^{(r)}_i,
\]

where \( \tilde{\mathbf{u}}_{\tau i} \in \mathbb{R}^p, \mathbf{B}^{(r)}_i \in [0,1]^K, \mathcal{D}^{(r)}_i \in \mathbb{R}^{K \times p}, \mathcal{F}^{(r)} \in \mathbb{R}^{p \times K}, i = 1, \ldots, M_P, j = 1, \ldots, M_C \).

It is obvious that there are many replicate elements in the product of \( \mathcal{D}^{(r)}_j \mathcal{F}^{(r)} \), after the parameter quantization. Therefore, it is unwise to compute the products in a one-by-one style. Instead, we first compute the result of the product \( \mathcal{D}^{(r)}_j \mathcal{F}^{(r)} \), i.e., constructing the lookup table, as follows:

\[
\mathcal{L}^{(r)}_j = \mathcal{D}^{(r)}_j \mathcal{F}^{(r)},
\]

where \( \mathcal{L}^{(r)}_j \in \mathbb{R}^{K \times K}, j = 1, \ldots, M_C, \tau = 1, \ldots, P \).

Then, in the application, we can look up the precomputed table instead of repeatedly computing which raises computational speed significantly. Hence, we can rewrite equation (19) as follows:

\[
\tilde{\mathbf{u}}_{\tau i} \approx \sum_{i=1}^{M_P} \mathbf{B}^{(r)}_i \mathcal{L}^{(r)}_j \mathcal{F}^{(r)} \mathbf{v}^{(r)}_i = \sum_{i=1}^{M_P} \mathbf{B}^{(r)}_i \mathcal{L}^{(r)}_j \mathbf{v}^{(r)}_i,
\]

where the product \( \mathbf{B}^{(r)}_i \mathcal{L}^{(r)}_j \mathbf{v}^{(r)}_i \) can be considered as the process of looking up the precomputed table \( \mathcal{L}^{(r)}_j \) instead of the matrix multiplication operation.

According to the mechanism of capsule network, the input vector \( \mathbf{z}_j \) of class layer capsule \( j \) can be computed by

\[
\mathbf{z}_j = \sum_{i=1}^{M_C} \theta_{ij} \tilde{\mathbf{u}}_{\tau i}, \quad j = 1, \ldots, M_C,
\]

where \( \theta_{ij} \) denotes the coupling coefficient determined by the iterative dynamic routing process (see Table 1). The routing part is actually a weighted sum of \( \tilde{\mathbf{u}}_{\tau i} \) with the coupling coefficient. The output vector of class layer capsule \( j \) is calculated by applying a nonlinear squashing function that can ensure short vectors to be shrunk to almost zero length, and long vectors get shrunk to a length slightly below one as

\[
\rho_j = \frac{\| \mathbf{z}_j \|^2}{1 + \| \mathbf{z}_j \|^2},
\]

where \( \rho_j \) denotes the output vector of class layer capsule \( j \).

Obviously, the capsule’s activation function actually suppresses and redistributes vector lengths. Its output can be used as the probability of the entity represented by the current class capsule. The quantized dynamic routing algorithm is shown in Table 1.

We construct the whole loss function of the proposed CWTACaps by integrating the margin loss [14], reconstruction loss [14], and the quantization loss as follows:

\[
\mathcal{L} = \mathcal{L}_{\text{max}} + \lambda_{\text{re}} \mathcal{L}_{\text{re}} + \lambda_{\text{qu}} \mathcal{L}_{\text{qu}},
\]

where \( \lambda_{\text{re}} \) and \( \lambda_{\text{qu}} \) denote positive coefficients and \( \mathcal{L}_{\text{max}}, \mathcal{L}_{\text{re}}, \) and \( \mathcal{L}_{\text{qu}} \) denote the margin loss function, the reconstruction loss function, and the quantization loss function, respectively. They are defined by equations (25)–(27) as follows:

\[
\mathcal{L}_{\text{max}} = T_c \max(0, e^x - \rho_j^2) + \eta_{\text{max}} \left(1 - T_c\right) \max(\rho_j - \epsilon)^2,
\]

\[
\mathcal{L}_{\text{re}} = X - X^2_F,
\]
Algorithm 1: Quantized dynamic routing.

Input: $B_i^{(r)}$, $\omega_j^{(r)}$, $\varphi_j^{(r)}$, $i$, $j$, $t$
Output: $\rho_j$

1. compute $\theta_{ij}$ using the quantized version of equation (21)
2. for all capsule $i$ in the primary layer and capsule $j$ in class layer: $b_{ij} \leftarrow 0$
3. for $t$ iterations do
   4. for all capsule $i$ in the primary layer and capsule $j$ in class layer: $\theta_{ij} \leftarrow \text{softmax}(b_{ij})$
   5. for all capsule $j$ in the class layer: compute the input vector $x_j$ using equation (22)
   6. for all capsule $j$ in the class layer: compute the output vector $\rho_j$ using equation (23)
   7. for all capsule $i$ in the primary layer and capsule $j$ in the class layer: $b_{ij} \leftarrow b_{ij} + \theta_{ij} \rho_j$
   8. Return $\rho_j$

\[ L_{qu} = \sum_{r=1}^{R} \sum_{j=1}^{M_j} B_j^{(r)} \mathbb{D}_j^{(r)} - \omega_{Fj}^2 + \eta_{qu} \sum_{r=1}^{R} \varphi_j^{(r)} - \mathbb{u}_{Fj}^{(r2)}, \]  

(27)

where $T_{c} = 1$ if correct classification, $\epsilon^+ = 0.9$ and $\epsilon^- = 0.1$, $\eta_{\text{max}}$ and $\eta_{\text{qu}}$ denote positive coefficients, usually selected as 0.5, and $\mathbb{X}$ denotes the reconstructed image.

3. Experiments

The aim of this section is to validate our proposed CWTACapsNet on three datasets: CUReT [40], DTD [41], and KTH-TIPS2-b [42]. For the CUReT dataset, we use the same subset as in [43], which contains 61 texture classes (92 images per class). The DTD dataset contains 47 classes (120 images per class). Besides CWTACapsNet, five state-of-the-art methods, T-CNN [8], SI-LGvMSP [1], Wavelet CNNs [11], and CapsNets [44], are employed for performance comparison

Models in experiments are trained under Ubuntu 16.04 with i7-8700 CPU, 64G RAM, and GeForce GTX Titan-XP GPU, and our proposed CWTACapsNet is deployed on Jetson TX2. To provide a direct comparison with published results, parameters of five state-of-the-art methods are set according to previous studies [1, 8, 11, 13, 44]. We use an exponential decay learning policy, with an initial learning rate of 0.001, 2000 decay steps, and 0.96 decay rate. We employ Adam optimizer to adjust the weights of CWTACapsNet in the training process. The batch size is set as 32. We implement data augmentation through rotating images with a random angle between 0° and 90°. We use 3 routing iterations to update capsule parameters in CWTACapsNet. The number of wavelet level in CWTACapsNet is selected according to the tradeoff between validation accuracy and network parameter amount. We thus choose 3-level wavelet decomposition. The learnable coefficient $\alpha$ is selected as 0.1, and $\lambda_{w}$ and $\lambda_{qu}$ are selected as 0.001 and 0.0013, respectively.

Table 2 illustrates classification accuracies and standard derivations of six methods. Table 2 indicates that CWTACapsNet achieves the best performance and is more stable than other methods. The tensor attention block makes CWTACapsNet be able to capture multidirectional dependencies while other methods cannot. FV-CNN performs better than CapsNets. FV-CNN and CapsNets both deal with pooling operation, and FV-CNN has some specific design to capture texture information. CNN-based texture classification methods tend to be limited by the lack of diversity of convolution filters. The multilevel wavelet decomposition extends both spatial and frequency features, which raises diversity of convolution filters and improves performance.

We add 10% white noise into texture datasets to evaluate robustness. Table 3 shows the performance of noisy datasets. Figure 5 shows accuracy for pure and noisy data. Figure 6 shows accuracy standard derivations (std) for pure and noisy data.

From Table 3 and Figures 5 and 6, we can find that CWTACapsNet achieves the best accuracy and robustness. Although CapsNets and CWTACapsNet are both based on capsule layer, CWTACapsNet significantly outperforms CapsNets. The memory requirement of CapsNets in the experiments is about 272M, while our proposed CWTACapsNet only requires 23.2M with about 10× speed-up. CWTACapsNet can be deployed and run on Jetson TX2, while CapsNets requires too much resource that Jetson TX2 hardly supported. The superiority of CWTACapsNet relies on three factors: the
multilevel wavelet decomposition extends features from spatial space to frequency space, the tensor attention block explores relationships from all possible directions and captures the dependencies cross channels, and the quantized dynamic routing significantly reduces memory requirement. Experimental results validate the effectiveness of CWTACapsNet.

Figure 5: The accuracy of six texture classification methods for pure and noisy texture datasets, i.e., (a) CUReT, (b) DTD, and (c) KTH-TIPS2-b.
4. Conclusion

In order to make capsule network efficiently explore spatial and spectral features and capture multidirectional channel dependencies, this paper proposes a novel capsule network named compressed wavelet tensor attention capsule network (CWTACapsNet). In CWTACapsNet, the compressed multiscale wavelet transform is designed to extract multiscale spectral features in frequency domain; the tensor attention blocks utilize matrization to capture multiple directional dependencies across convolutional channels in terms of each scale information; furthermore, we propose quantized dynamic routing process for speeding up and storage reduction. Experimental studies have shown that the proposed CWTACapsNet provides the best performance on both classification result and antinoise robustness; moreover, CWTACapsNet significantly reduces the computational and storage complexities. In the future, we will incorporate parallel computation methods into CWTACapsNet to further improve efficiency.

Data Availability

The authors approve that data used to support the finding of this study are publicly available. The datasets can be achieved from the links provided by [40–42]. CUReT Dataset is available at https://www.cs.columbia.edu/CAVE/software/curet/html/download.h

Conflicts of Interest

The authors declare that they have no conflicts of interest.
Acknowledgments

This work was supported by the Chinese National Natural Science Foundations (nos. 61972104 and 61571141).

References

[1] N. Alpaslan and K. Hanbay, “Multi-scale shape index-based local binary patterns for texture classification,” *IEEE Signal Processing Letters*, vol. 27, pp. 660–664, 2020.

[2] Q. Kou, D. Cheng, H. Zhuang, and R. Gao, “Cross-complementary local binary pattern for robust texture classification,” *IEEE Signal Processing Letters*, vol. 26, no. 1, pp. 129–133, 2019.

[3] S. S. Moghadasian and S. Gazor, “Sparsely localized time-frequency energy distributions for multi-component LFM signals,” *IEEE Signal Processing Letters*, vol. 27, pp. 6–10, 2020.

[4] X. Dong, H. Zhou, and J. Dong, “Texture classification using pair-wise difference pooling-based bilinear convolutional neural networks,” *IEEE Transactions on Image Processing*, vol. 29, pp. 8776–8790, 2020.

[5] L. Liu, J. Chen, P. Fieguth, G. Zhao, R. Chellappa, and M. Pietikäinen, “From BoW to CNN: two decades of texture representation for texture classification,” *International Journal of Computer Vision*, vol. 127, no. 1, pp. 74–109, 2019.

[6] A. Humeau-Heurtier, “Texture feature extraction methods: a survey,” *IEEE Access*, vol. 7, pp. 8975–9000, 2019.

[7] X. Dong, J. Dong, and M. Chandler, “Perceptual texture similarity estimation: an evaluation of computational features,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1, 2020.

[8] M. Cimpoi, S. Maji, and A. Vedaldi, “Deep filter banks for texture recognition and segmentation,” in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3828–3836, Boston, MA, USA, June 2015.

[9] X. Dong and M. J. Chandler, “Perceptually motivated image features using contours,” *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5050–5062, 2016.

[10] Y. Kim, B. Ham, M. N. Do, and K. Sohn, “Structure-texture image decomposition using deep variational priors,” *IEEE Transactions on Image Processing*, vol. 28, no. 6, pp. 2692–2704, 2019.

[11] S. Fujieda, K. Takayama, and T. Hachisuka, “Wavelet convolutional neural networks for texture classification,” 2017, http://arxiv.org/abs/1707.07394.

[12] M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi, “Deep filter banks for texture recognition, description, and segmentation,” *International Journal of Computer Vision*, vol. 118, no. 1, pp. 65–94, 2016.

[13] V. Andrearczyk and P. F. Whelan, “Using filter banks in convolutional neural networks for texture classification,” *Pattern Recognition Letters*, vol. 84, pp. 63–69, 2016.

[14] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” in *Proceedings of the 2017 The Thirty-First Annual Conference on Neural Information Processing Systems*, pp. 3856–3866, Long Beach, CA, USA, May 2017.

[15] D. Xu, Z. Tian, R. Lai, X. Kong, Z. Tan, and W. Shi, “Deep learning based emotion analysis of microblog texts,” *Information Fusion*, vol. 64, pp. 1–11, 2020.

[16] C. Xiang, L. Zhang, Y. Tang, W. Zou, and C. Xu, “MS-capsnet: a novel multi-scale capsule network,” *IEEE Signal Processing Letters*, vol. 25, no. 12, pp. 1850–1854, 2018.

[17] S. Zhang, W. Zhao, X. Wu, and Q. Zhou, “Fast dynamic routing based on weighted kernel density estimation,” *Concurrency and Computation: Practice and Experience*, vol. 5281, pp. 1–10, 2019.

[18] D. Guo, H. Wang, S. Wang, and M. Wang, “Textual-visual reference-aware attention network for visual dialog,” *IEEE Transactions on Image Processing*, vol. 29, pp. 6655–6666, 2020.

[19] T. Xu, “Design of English diagnostic practice sentence repetition recognition system based on matching tree and edge computing,” *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6653145, 8 pages, 2021.

[20] Z. Xie, L. Hu, Y. Huang, and J. Pang, “A semiopportunistic task allocation framework for mobile crowdsensing with deep learning,” *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6643229, 15 pages, 2021.

[21] M. Shaﬁq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, “CorrAUC: a malicious bot-IoT traffic detection method in IoT network using machine-learning techniques,” *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3242–3254, 2021.

[22] M. Shaﬁq, Z. Tian, Y. Sun, X. Du, and M. Guizani, “Selection of effective machine learning algorithm and Bot-IOI attacks trafﬁc identiﬁcation for internet of things in smart city,” *Future Generation Computer Systems*, vol. 107, pp. 433–442, 2020.

[23] M. Shaﬁq, Z. Tian, A. K. Bashir, A. Jolfaei, and X. Yu, “Data mining and machine learning methods for sustainable smart cities trafﬁc classiﬁcation: a survey,” *Sustainable Cities and Society*, vol. 60, Article ID 102177, 2020.

[24] M. Shaﬁq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, “IoT malicious trafﬁc identiﬁcation using wrapper-based feature selection mechanisms,” *Computers & Security*, vol. 94, Article ID 101863, 2020.

[25] M. Shaﬁq, Z. Tian, A. K. Bashir, K. Cengiz, and A. Tahir, “SoftSystem: smart edge computing device selection method for IoT based on soft set technique,” *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 8864301, 2020.

[26] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, “A survey on access control in the age of internet of things,” *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4682–4696, 2020.

[27] M. Li, Y. Sun, H. Lu, S. Maharjan, and Z. Tian, “Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6266–6278, 2020.

[28] S. Su, Z. Tian, S. Liang, S. Li, S. Du, and N. Guizani, “A reputation management scheme for efficient malicious vehicle identiﬁcation over 5G networks,” *IEEE Wireless Communications*, vol. 27, no. 3, pp. 46–52, 2020.

[29] Z. Gu, L. Wang, X. Chen et al., “Epidemic risk assessment by a novel communication station based method,” *IEEE Transactions on Network Science and Engineering*, p. 1, 2021.

[30] Y. Wang, Z. Tian, Y. Sun, X. Du, and N. Guizani, “Locjury: an IBN-based location privacy preserving scheme for IoCV,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2020.

[31] Y. Wang, Z. Tian, H. Zhang, S. Su, and W. Shi, “A privacy preserving scheme for nearest neighbor query,” *Sensors (Basel, Switzerland)*, vol. 18, no. 8, p. 2440, 2018.

[32] W. Jiang, Z. Tian, H. Zhang, and X. Song, “A stochastic game theoretic approach to attack prediction and optimal active defense strategy decision,” in *Proceedings of the 2008 IEEE International Conference on Networking, Sensing and Control*, pp. 648–653, Sanya, China, April 2008.

[33] W. Jiang, B. Fang, H. Zhang, Z. Tian, and X. Song, “Optimal network security strengthening using attack-defense game model,” in *Proceedings of the 2009 Sixth International
Conference on Information Technology: New Generations, pp. 475–480, Las Vegas, NV, USA, April 2009.

[34] S. G. Mallat, “A theory for multiresolution signal decomposition: the wavelet representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[35] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, “Multi-level wavelet-CNN for image restoration,” in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 886–88609, Salt Lake City, UT, USA, 2018.

[36] F. Babiloni, I. Marras, G. Slabaugh, and S. Zafeiriou, “TESA: tensor element self-attention via matricization,” in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13942–13951, Seattle, WA, USA, June 2020.

[37] Y. Xiong, Z. Zeng, R. Chakraborty et al., “Nyströmformer: a nyström-based algorithm for approximating self-attention,” 2021, http://arxiv.org/abs/2102.03902.

[38] S. Wang, A. Gittens, and M. W. Mahoney, “Scalable kernel K-means clustering with Nyström approximation: relative-error bounds,” *Journal of Machine Learning Research*, vol. 20, no. 12, pp. 1–49, 2019.

[39] J. Cheng, J. Wu, C. Leng, Y. Wang, and Q. Hu, “Quantized CNN: a unified approach to accelerate and compress convolutional networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4730–4743, 2018.

[40] M. Varma and A. Zisserman, “A statistical approach to material classification using image patch exemplars,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 11, pp. 2032–2047, 2009.

[41] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, “Describing textures in the wild,” in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3606–3613, Columbus, OH, USA, June 2014.

[42] E. Hayman, B. Caputo, M. Fritz, and J.-O. Eklundh, “On the significance of real-world conditions for material classification,” in *Proceedings of the 2004 8th European Conference on Computer Vision*, pp. 253–266, Prague, Czech Republic, May 2004.

[43] L. Liu and P. W. Fieguth, “Texture classification from random features,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 574–586, 2012.

[44] B. Mamidibathula, S. Amirneni, S. S. Sistla, and N. Patnam, “Texture classification using capsule networks pattern recognition and image analysis,” *Pattern Recognition and Image Analysis*, vol. 11867, pp. 589–599, 2019.