

DESCRIPTION

TITLE OF INVENTION: DATA PROCESSING DEVICE, DATA PROCESSING METHOD, AND STORAGE MEDIUM

TECHNICAL FIELD

[0001] The invention relates to a data processing device and a data processing method that encode and compress information about a configuration of a neural network, and relates to a storage medium storing compressed data.

BACKGROUND ART

[0002] As a method for solving classification problems and regression problems of input data, there is machine learning.

For the machine learning, there is a technique called a neural network that imitates a brain's neural circuit (neurons). In the neural network, classification (discrimination) or regression of input data is performed using a probabilistic model (a discriminative model or a generative model) represented by a network in which neurons are mutually connected to each other.

Furthermore, in a convolutional neural network which is a neural network having not only a fully-connected layer but also a convolution layer and a pooling layer, a network can be created that implements data processing other than classification and regression, such as a network that implements a data filtering process. For example, a convolutional neural network can implement an image or audio filtering process that achieves noise removal of an input signal, an improvement in quality, or the like, with an image or audio being an input, a high-frequency restoration process for audio with missing high frequencies such as compressed audio, inpainting for an image whose

region is partially missing, a super-resolution process for an image, etc.

In addition to the above, in recent years, there has also been released a new neural network called a generative adversarial network in which the network is constructed by combining a generative model and a discriminative model that are used to determine whether data is real by inputting the data to the discriminative model that determines whether data generated by the generative model is real data (whether the data is not data generated by the generative model), and the generative model is adversarially trained in such a manner that the discriminative model cannot distinguish generated data as generated data, and the discriminative model is adversarially trained so as to be able to distinguish generated data as generated data, by which creation of a generative model with high accuracy is implemented.

[0003] In these neural networks, by optimizing network parameters by training using a large amount of data, an improvement in performance can be achieved.

Note, however, that there is a tendency that the data size of a neural network becomes large, and computational load on a computer using the neural network also increases.

[0004] Regarding this, Non-Patent Literature 1 describes a technique for scalar-quantizing and encoding edge weights which are parameters of a neural network. By scalar-quantizing and encoding the edge weights, the data size of data about edges is compressed.

CITATION LIST

PATENT LITERATURES

[0005] Non-Patent Literature 1: Vincent Vanhoucke, Andrew Senior, Mark Z. Mao, "Improving the speed of neural networks on CPUs," Proc. Deep Learning and

Unsupervised Feature Learning NIPS Workshop, 2011.

SUMMARY OF INVENTION

TECHNICAL PROBLEM

[0006] However, the optimal values of weights assigned to a respective plurality of edges in a neural network vary depending on network training results and are not fixed.

Hence, variations occur in the compression size of edge weights, and the technique described in Non-Patent Literature 1 has a problem that high compression of parameter data about edges of a neural network cannot be achieved.

[0007] The invention is to solve the above-described problem, and an object of the invention is to obtain a data processing device and a data processing method that can highly compress parameter data of a neural network, and a storage medium storing compressed data.

SOLUTION TO PROBLEM

[0008] A data processing device according to the invention includes a data processing unit, a compression controlling unit, and an encoding unit. The data processing unit processes input data using a neural network. The compression controlling unit determines quantization steps and generates quantization information that defines the quantization steps, the quantization steps being used when parameter data of the neural network is quantized. The encoding unit encodes network configuration information and the quantization information to generate compressed data, the network configuration information including the parameter data quantized using the quantization steps determined by the compression controlling unit.

ADVANTAGEOUS EFFECTS OF INVENTION

[0009] According to the invention, quantization information that defines quantization steps which are used when parameter data of a neural network is quantized, and network configuration information including the parameter data quantized using the quantization steps in the quantization information are encoded to generate compressed data.

Accordingly, the parameter data of the neural network can be highly compressed.

By using quantization information and network configuration information which are decoded from the compressed data, a neural network optimized on the encoding side can be constructed on the decoding side.

BRIEF DESCRIPTION OF DRAWINGS

[0010] FIG. 1 is a block diagram showing a configuration of a data processing device (encoder) according to a first embodiment of the invention.

FIG. 2 is a block diagram showing a configuration of a data processing device (decoder) according to the first embodiment.

FIG. 3A is a block diagram showing a hardware configuration that implements the functions of a data processing device according to the first embodiment, and FIG. 3B is a block diagram showing a hardware configuration that executes software that implements the functions of the data processing device according to the first embodiment.

FIG. 4 is a flowchart showing the operation of the data processing device (encoder) according to the first embodiment.

FIG. 5 is a flowchart showing the operation of the data processing device (decoder) according to the first embodiment.

FIG. 6 is a diagram showing an exemplary configuration of a neural network in the first embodiment.

FIG. 7 is a diagram showing an example of a convolution process for one-dimensional data in the first embodiment.

FIG. 8 is a diagram showing an example of a convolution process for two-dimensional data in the first embodiment.

FIG. 9 is a diagram showing a matrix of node-by-node edge weight information in an lth layer of a neural network.

FIG. 10 is a diagram showing a matrix of quantization steps for the node-by-node edge weight information in the lth layer of the neural network.

FIG. 11 is a diagram showing a matrix of edge weight information in a convolutional layer.

FIG. 12 is a diagram showing a matrix of quantization steps for the edge weight information in the convolutional layer.

FIG. 13 is a diagram showing syntax of information included in quantization information in the first embodiment.

FIG. 14 is a diagram showing syntax for each matrix of information included in the quantization information in the first embodiment.

FIG. 15 is a diagram showing syntax for each layer of information included in the quantization information in the first embodiment.

DESCRIPTION OF EMBODIMENTS

[0011] To more specifically describe the invention, modes for carrying out the invention will be described below with reference to the accompanying drawings.
First Embodiment.

FIG. 1 is a block diagram showing a configuration of a data processing device 100 according to a first embodiment of the invention. In FIG. 1, the data processing

device 100 processes input data using a trained neural network, and outputs processing results.

In addition, the data processing device 100 functions as an encoder that encodes quantization information and network configuration information, and includes a data processing unit 101, a compression controlling unit 102, and an encoding unit 103.

[0012] The data processing unit 101 processes input data using the above-described neural network.

In addition, the data processing unit 101 accepts, as input, quantization information generated by the compression controlling unit 102, and quantizes parameter data of the neural network using quantization steps defined in the quantization information. Then, the data processing unit 101 outputs network configuration information including the above-described quantized parameter data to the encoding unit 103.

[0013] For the above-described neural network used by the data processing unit 101, a predetermined neural network may be used, or the parameter data may be optimized by training.

When parameter data of a neural network is trained, neural network training is performed on the neural network in a predetermined initial state (initial values of the parameter data) using input data to be trained, and then quantization information generated by the compression controlling unit 102 is inputted, and the parameter data of the neural network is quantized using quantization steps defined in the quantization information.

Then, with the quantized neural network being in an initial state for the next training, the above-described training and quantization are performed. A neural

network obtained as a result of repeating the training and quantization processes L times (L is an integer greater than or equal to one) is outputted, as a part of network configuration information, to the encoding unit 103.

Note that when $L = 1$, the quantized neural network is not retrained and thus it can be said that this is the same process as using a neural network that is trained outside without being trained by the data processing unit 101. That is, the difference is only whether training is performed by the data processing unit 101 or outside.

[0014] The network configuration information is information indicating a configuration of the neural network, and includes, for example, the number of network layers, the number of nodes for each of the layers, edges that link nodes, weight information assigned to each of the edges, activation functions representing outputs from the nodes, and type information for each of the layers (e.g., a convolutional layer, a pooling layer, or a fully-connected layer).

The parameter data of the neural network includes, for example, weight information assigned to edges that connect nodes of the neural network.

[0015] The compression controlling unit 102 determines quantization steps which are used when the parameter data of the neural network is quantized, and generates quantization information that defines the quantization steps.

For example, the compression controlling unit 102 determines quantization steps that change on an edge-by-edge, node-by-node, kernel-by-kernel, or layer-by-layer basis in the neural network.

[0016] The quantization information is information that defines quantization steps which are used when the parameter data of the neural network is quantized. The quantization step is a width (quantization width) used when parameter data is quantized, and the larger the quantization step is, the more coarsely the parameter data is broken

down, and thus, the compression ratio increases. The smaller the quantization step is, the more finely the parameter data is broken down, and thus, the compression ratio decreases.

Specifically, a quantized value k is represented by the following equation (1):

$$k = \text{floor}((x/Q) + d_0) + d_1 \quad (1)$$

In the above-described equation (1), x represents the value of a parameter to be quantized, Q represents a quantization step, d_0 ($0 \leq d_0 < 1$) represents an adjustment offset in a range of values to be quantized that correspond to respective quantized values, d_1 ($0 \leq d_1 < 1$) represents an offset that adjusts the quantized value, and $\text{floor}()$ represents a round-down processing function. Furthermore, there is also a method for providing a dead zone in which quantized values k for values x to be quantized within a range set in the above-described equation (1) are 0.

In addition, the value y of a quantized parameter is as shown in the following equation (2):

$$y = kQ \quad (2)$$

When the minimum change unit of the quantization steps described above is changed from units of edges to units of layers, the compression ratio of parameter data by quantization increases, and thus, pre-encoding parameter data can be reduced.

[0017] The encoding unit 103 encodes the network configuration information including the parameter data quantized by the data processing unit 101 and the quantization information generated by the compression controlling unit 102, to generate compressed data.

Note that the network configuration information inputted to the encoding unit 103 from the data processing unit 101 is network configuration information including the parameter data which is quantized by the data processing unit 101 using the

quantization steps determined by the compression controlling unit 102.

[0018] FIG. 2 is a block diagram showing a configuration of a data processing device 200 according to the first embodiment. In FIG. 2, the data processing device 200 processes input data using a neural network obtained by decoding compressed data, and outputs processing results. The processing results include classification results or regression analysis results of the input data, as with the data processing device 100.

The data processing device 200 functions as a decoder that decodes quantization information and network configuration information from compressed data, and includes a decoding unit 201 and a data processing unit 202.

[0019] The decoding unit 201 decodes quantization information and network configuration information from the compressed data encoded by the encoding unit 103 as described above.

The network configuration information decoded by the decoding unit 201 includes results (quantized values k) obtained by quantizing parameter data such as edge weight information, the edge weight information having been optimized as a result of performing training on a side of the data processing device 100 which is an encoder.

The quantization information decoded by the decoding unit 201 defines quantization steps Q which are used when the parameter data is quantized. From the above-described results k obtained by quantizing the parameter data and the above-described quantization steps Q , quantized parameters y are decoded in accordance with the above-described equation (2). These decoding results are outputted from the decoding unit 201 to the data processing unit 202.

[0020] The data processing unit 202 processes input data using a neural network.

In addition, the data processing unit 202 inversely quantizes the edge weight information which is parameter data, using the quantization information and network

configuration information decoded from the compressed data by the decoding unit 201.

Furthermore, the data processing unit 202 constructs a neural network using the network configuration information including the inversely quantized parameter data.

[0021] As such, the data processing unit 202 constructs a neural network including parameter data such as edge weight information which has been optimized as a result of performing training on the data processing device 100 side, using information decoded from compressed data, and processes input data using the neural network. Accordingly, the compression size of the optimized parameter data can be made uniform between the data processing device 100 and the data processing device 200, thereby being able to achieve high compression of the parameter data.

[0022] FIG. 3A is a block diagram showing a hardware configuration that implements the functions of the data processing device 100. In FIG. 3A, a processing circuit 300 is a dedicated circuit that functions as the data processing device 100. FIG. 3B is a block diagram showing a hardware configuration that executes software that implements the functions of the data processing device 100. In FIG. 3B, a processor 301 and a memory 302 are connected to each other by a signal bus.

[0023] The functions of the data processing unit 101, the compression controlling unit 102, and the encoding unit 103 in the data processing device 100 are implemented by a processing circuit.

Namely, the data processing device 100 includes a processing circuit for performing processes at step ST1 to ST3 which will be described later with reference to FIG. 4.

The processing circuit may be dedicated hardware, but may be a Central Processing Unit (CPU) that executes programs stored in a memory.

[0024] When the above-described processing circuit is the dedicated hardware shown

in FIG. 3A, the processing circuit 300 corresponds, for example, to a single circuit, a composite circuit, a programmed processor, a parallel programmed processor, an Application Specific Integrated Circuit (ASIC), a Field-Programmable Gate Array (FPGA), or a combination thereof.

Note that the functions of the data processing unit 101, the compression controlling unit 102, and the encoding unit 103 may be implemented by different respective processing circuits, or the functions may be collectively implemented by a single processing circuit.

[0025] When the above-described processing circuit is the processor shown in FIG. 3B, the functions of the data processing unit 101, the compression controlling unit 102, and the encoding unit 103 are implemented by software, firmware, or a combination of software and firmware.

The software or firmware is described as programs and stored in the memory 302.

The processor 301 implements the functions of the data processing unit 101, the compression controlling unit 102, and the encoding unit 103, by reading and executing the programs stored in the memory 302. Namely, the data processing device 100 includes the memory 302 for storing programs that when executed by the processor 301, cause the processes at step ST1 to ST3 shown in FIG. 4 to be consequently performed.

The programs cause a computer to perform procedures or methods of the data processing unit 101, the compression controlling unit 102, and the encoding unit 103.

The memory 302 may be a computer readable storage medium having stored therein programs for causing a computer to function as the data processing unit 101, the compression controlling unit 102, and the encoding unit 103.

[0026] The memory 302 corresponds, for example, to a nonvolatile or volatile semiconductor memory such as a Random Access Memory (RAM), a Read Only Memory (ROM), a flash memory, an Erasable Programmable Read Only Memory (EPROM), or an Electrically-EPROM (EEPROM), a magnetic disk, a flexible disk, an optical disc, a compact disc, a MiniDisc, or a DVD.

[0027] Note that some of the functions of the data processing unit 101, the compression controlling unit 102, and the encoding unit 103 may be implemented by dedicated hardware, and some of the functions may be implemented by software or firmware.

For example, the function of the data processing unit 101 may be implemented by a processing circuit which is dedicated hardware, and the functions of the compression controlling unit 102 and the encoding unit 103 may be implemented by the processor 301 reading and executing programs stored in the memory 302.

As such, the processing circuit can implement the above-described functions by hardware, software, firmware, or a combination thereof.

[0028] Note that although the data processing device 100 is described, the same also applies to the data processing device 200. For example, the data processing device 200 includes a processing circuit for performing processes at step ST1a to ST4a which will be described later with reference to FIG. 5. The processing circuit may be dedicated hardware, but may be a CPU that executes programs stored in a memory.

[0029] When the above-described processing circuit is the dedicated hardware shown in FIG. 3A, the processing circuit 300 corresponds, for example, to a single circuit, a composite circuit, a programmed processor, a parallel programmed processor, an ASIC, an FPGA, or a combination thereof.

Note that the functions of the decoding unit 201 and the data processing unit

202 may be implemented by different respective processing circuits, or the functions may be collectively implemented by a single processing circuit.

[0030] When the above-described processing circuit is the processor shown in FIG. 3B, the functions of the decoding unit 201 and the data processing unit 202 are implemented by software, firmware, or a combination of software and firmware.

The software or firmware is described as programs and stored in the memory 302.

The processor 301 implements the functions of the decoding unit 201 and the data processing unit 202, by reading and executing the programs stored in the memory 302.

Namely, the data processing device 200 includes the memory 302 for storing programs that when executed by the processor 301, cause the processes at step ST1a to ST4a shown in FIG. 5 to be consequently performed.

The programs cause a computer to perform procedures or methods of the decoding unit 201 and the data processing unit 202.

The memory 302 may be a computer readable storage medium having stored therein programs for causing a computer to function as the decoding unit 201 and the data processing unit 202.

[0031] Note that some of the functions of the decoding unit 201 and the data processing unit 202 may be implemented by dedicated hardware, and some of the functions may be implemented by software or firmware.

For example, the function of the decoding unit 201 may be implemented by a processing circuit which is dedicated hardware, and the function of the data processing unit 202 may be implemented by the processor 301 reading and executing a program stored in the memory 302.

[0032] Next operation will be described.

FIG. 4 is a flowchart showing the operation of the data processing device 100.

The following describes a case in which parameter data of a neural network is edge weight information.

The compression controlling unit 102 determines a quantization step which is used when weight information of each of a plurality of edges included in a trained neural network is quantized, and generates quantization information that defines the quantization step (step ST1). The quantization information is outputted from the compression controlling unit 102 to the data processing unit 101 and the encoding unit 103.

[0033] When the data processing unit 101 accepts, as input, the quantization information from the compression controlling unit 102, the data processing unit 101 quantizes the above-described edge weight information of the neural network using the quantization step in the quantization information (step ST2). The data processing unit 101 generates network configuration information including the quantized edge weight information, and outputs the network configuration information to the encoding unit 103.

[0034] The encoding unit 103 encodes the above-described network configuration information inputted from the data processing unit 101 and the above-described quantization information inputted from the compression controlling unit 102 (step ST3).

Compressed data of the above-described network configuration information and quantization information encoded by the encoding unit 103 is outputted to the data processing device 200.

[0035] FIG. 5 is a flowchart showing the operation of the data processing device 200.

The decoding unit 201 decodes quantization information and network

configuration information from the above-described compressed data encoded by the encoding unit 103 (step ST1a). The quantization information and the network configuration information are outputted from the decoding unit 201 to the data processing unit 202.

[0036] Then, the data processing unit 202 calculates edge weight information which is inversely quantized using the quantization information and network configuration information decoded from the compressed data by the decoding unit 201 (step ST2a).

[0037] Subsequently, the data processing unit 202 constructs a neural network using the network configuration information including the inversely quantized edge weight information (step ST3a).

Accordingly, the data processing device 200 can construct the neural network trained by the data processing device 100.

[0038] The data processing unit 202 processes input data using the neural network constructed at step ST3a (step ST4a).

[0039] FIG. 6 is a diagram showing an exemplary configuration of a neural network in the first embodiment.

In the neural network shown in FIG. 6, input data ($x_1, x_2, \dots, x_{N1}$) is processed by each layer, and processing results ($y_1, \dots, y_{NL}$) are outputted.

In FIG. 6, N_I ($I = 1, 2, \dots, L$) represents the number of nodes in an I th layer, and L represents the number of layers in the neural network.

As shown in FIG. 6, the neural network includes an input layer, hidden layers, and an output layer, and is structured in such a manner that in each of these layers a plurality of nodes are linked by edges.

An output value of each of the plurality of nodes can be calculated, from an output value of a node in the immediately previous layer which is linked to the

corresponding one of the plurality of nodes by an edge, weight information of the edge, and an activation function set on a layer-by-layer basis.

[0040] An example of a neural network includes a Convolutional Neural Network (CNN). In hidden layers of the CNN, convolutional layers and pooling layers are alternately linked to each other, and one or more fully-connected neural network layers (fully-connected layers) are provided in accordance with final outputs. For activation functions for the convolutional layers, for example, ReLU functions are used.

Note that a network called a deep neural network (DNN) (which is also called deep learning, a deep CNN (DCNN), etc.) is a network including multiple CNN layers.

[0041] FIG. 7 is a diagram showing an example of a convolution process for one-dimensional data in the first embodiment, and shows a convolutional layer that performs a convolution process for one-dimensional data. The one-dimensional data includes, for example, audio data and time-series data.

The convolutional layer shown in FIG. 7 include nine nodes 10-1 to 10-9 in a previous layer and three nodes 11-1 to 11-3 in a subsequent layer.

Edges 12-1, 12-6, and 12-11 are each assigned an identical weight, edges 12-2, 12-7, and 12-12 are each assigned an identical weight, edges 12-3, 12-8, and 12-13 are each assigned an identical weight, edges 12-4, 12-9, and 12-14 are each assigned an identical weight, and edges 12-5, 12-10, and 12-15 are each assigned an identical weight. In addition, the weights of the edges 12-1 to 12-5 may all have different values, or some or all of the weights may have the same value.

[0042] Of the nine nodes 10-1 to 10-9 in the previous layer, five nodes are linked to one node in the subsequent layer with the above-described weights. The kernel size K is five, and the kernel is defined by a combination of these weights.

For example, as shown in FIG. 7, the node 10-1 is linked to a node 11-1

through the edge 12-1, the node 10-2 is linked to the node 11-1 through the edge 12-2, the node 10-3 is linked to the node 11-1 through the edge 12-3, the node 10-4 is linked to the node 11-1 through the edge 12-4, and the node 10-5 is linked to the node 11-1 through the edge 12-5. The kernel is defined by a combination of the weights of the edges 12-1 to 12-5.

[0043] The node 10-3 is linked to a node 11-2 through the edge 12-6, the node 10-4 is linked to the node 11-2 through the edge 12-7, the node 10-5 is linked to the node 11-2 through the edge 12-8, the node 10-6 is linked to the node 11-2 through the edge 12-9, and the node 10-7 is linked to the node 11-2 through the edge 12-10. The kernel is defined by a combination of the weights of the edges 12-6 to 12-10.

The node 10-5 is linked to a node 11-3 through the edge 12-11, the node 10-6 is linked to the node 11-3 through the edge 12-12, the node 10-7 is linked to the node 11-3 through the edge 12-13, the node 10-8 is linked to the node 11-3 through the edge 12-14, and the node 10-9 is linked to the node 11-3 through the edge 12-15. The kernel is defined by a combination of the weights of the edges 12-11 to 12-15.

[0044] In a process for input data using a CNN, the data processing unit 101 or the data processing unit 202 performs, for each kernel, a convolution operation at an interval of the number of steps S (in FIG. 7, $S = 2$) using a corresponding combination of edge weights of a convolutional layer. The combination of edge weights is determined for each kernel by training.

Note that in a CNN used for image recognition, the network is often constructed using a convolutional layer including a plurality of kernels.

[0045] FIG. 8 is a diagram showing an example of a convolution process for two-dimensional data in the first embodiment, and shows a convolution process for two-dimensional data such as image data.

In the two-dimensional data shown in FIG. 8, a kernel 20 is a block region with a size of K_x in an x-direction and a size of K_y in a y-direction. The kernel size K is represented by $K = K_x \times K_y$.

In the two-dimensional data, the data processing unit 101 or the data processing unit 202 performs a convolution operation on data for each kernel 20 at an interval of the number of steps S_x in the x-direction and an interval of the number of steps S_y in the y-direction. Here, the steps S_x and S_y are integers greater than or equal to one.

[0046] FIG. 9 is a diagram showing a matrix of node-by-node edge weight information in an l th layer ($l = 1, 2, \dots, L$) which is a fully-connected layer of a neural network.

FIG. 10 is a diagram showing a matrix of quantization steps for the node-by-node edge weight information in the l th layer ($l = 1, 2, \dots, L$) which is a fully-connected layer of the neural network.

[0047] In the neural network, a combination of weights w_{ij} for each layer shown in FIG. 9 is data for constructing the network. Hence, in a multi-layer neural network such as a DNN, the amount of data is generally several hundred Mbytes or more and a large memory size is also required. Note that i is the node index and $i = 1, 2, \dots, N_l$, and j is the edge index and $j = 1, 2, \dots, N_{l-1}$.

[0048] Hence, in order to reduce the amount of data of edge weight information, the data processing device 100 according to the first embodiment quantizes the weight information. As shown in FIG. 10, the quantization steps q_{ij} are set for the respective edge weights w_{ij} .

Furthermore, a common quantization step may be used among a plurality of node indices or a plurality of edge indices, or among a plurality of node indices and edge indices. By doing so, quantization information to be encoded can be reduced.

[0049] FIG. 11 is a diagram showing a matrix of edge weight information in a

convolutional layer.

FIG. 12 is a diagram showing a matrix of quantization steps for the edge weight information in the convolutional layer. In the convolutional layer, an edge weight for a single kernel is common to all nodes, and by reducing the number of edges connected per node, i.e., the kernel size K , a kernel region can be made small.

[0050] FIG. 11 shows data of edge weights $w_{ij'}$ set on a kernel-by-kernel basis, and FIG. 12 shows data of quantization steps $q_{ij'}$ set on a kernel-by-kernel basis.

Note that i' is the kernel index and $i' = 1, 2, \dots, M_l$ ($l = 1, 2, \dots, L$), and j' is the edge index and $j' = 1, 2, \dots, K_l$.

Furthermore, a common quantization step may be used among a plurality of kernel indices or a plurality of edge indices, or among a plurality of kernel indices and edge indices. By doing so, quantization information to be encoded can be reduced.

[0051] At step ST1 of FIG. 4, the compression controlling unit 102 determines quantization steps which are used in a weight quantization process performed by the data processing unit 101, and outputs the quantization steps as quantization information to the data processing unit 101. The quantization steps are the quantization steps q_{ij} shown in FIG. 10 and the quantization steps $q_{ij'}$ shown in FIG. 12.

[0052] At step ST2 of FIG. 4, the data processing unit 101 quantizes the edge weights w_{ij} shown in FIG. 9 using the quantization steps q_{ij} shown in FIG. 10, and outputs network configuration information including the quantized weights w_{ij} to the encoding unit 103.

Likewise, at step ST2 of FIG. 4, the data processing unit 101 quantizes the edge weights $w_{ij'}$ shown in FIG. 11 using the quantization steps $q_{ij'}$ shown in FIG. 12, and outputs network configuration information including the quantized weights $w_{ij'}$ to the encoding unit 103.

Note that the network configuration information includes, in addition to the quantized weights, the number of network layers, the number of nodes for each of the layers, edges that link nodes, weight information assigned to each of the edges, activation functions representing outputs from the nodes, type information for each of the layers (e.g., a convolutional layer, a pooling layer, or a fully-connected layer), etc. Note, however, that information that is fixed (defined) in advance between the data processing device 100 and the data processing device 200 is not included in network configuration information to be encoded.

[0053] FIG. 13 is a diagram showing syntax of information included in quantization information in the first embodiment.

FIG. 14 is a diagram showing syntax for each matrix of information included in the quantization information in the first embodiment.

FIG. 15 is a diagram showing syntax for each layer of information included in the quantization information in the first embodiment.

[0054] In FIG. 13, the flag "quant_enable_flag", the flag "layer_adaptive_quant_flag", the flag "matrix_adaptive_quant_flag", and the quantization step "fixed_quant_step" are encoding parameters of quantization information to be encoded by the encoding unit 103.

In addition, L is the number of layers.

[0055] In FIG. 14, the quantization step "base_quant_step[j]", the flag "prev_quant_copy_flag[i-1]", and the difference value "diff_quant_value[i-1][j]" are encoding parameters of quantization information to be encoded by the encoding unit 103.

In addition, C is the number of nodes $N_{\text{layer_id}}$ or the number of kernels $M_{\text{layer_id}}$. Furthermore, E is the number of edges $N_{\text{layer_id-1}}$ or the kernel size $K_{\text{layer_id}}$.

[0056] In FIG. 15, the quantization step "base_layer_quant_step", the flag "layer_quant_copy_flag[i-2]", and the quantization step "layer_quant_step[i-2]" are encoding parameters of quantization information to be encoded by the encoding unit 103. In addition, L is the number of layers.

[0057] The information shown in FIG. 13 includes the flag "quant_enable_flag" that sets whether to quantize edge weight information in the network.

When the flag "quant_enable_flag" is 0 (false), all edge weight information in the network is not quantized. Namely, no quantization steps are set in the quantization information.

On the other hand, when the flag "quant_enable_flag" is 1 (true), the compression controlling unit 102 refers to the flag "layer_adaptive_quant_flag".

[0058] When the flag "layer_adaptive_quant_flag" is 0 (false), the compression controlling unit 102 sets, in the quantization information, the quantization step "fixed_quant_step" which is common to all edges in the network.

When the flag "layer_adaptive_quant_flag" is 1 (true), the compression controlling unit 102 refers to the flag "matrix_adaptive_quant_flag".

[0059] When the flag "matrix_adaptive_quant_flag" is 0 (false), the compression controlling unit 102 determines a quantization step which is common on a per-layer basis, as a quantization step for weight information of each of a plurality of edges in the network.

Note, however, that an input layer (first layer) has no edges and thus no quantization steps are set for the input layer.

Note that FIG. 15 shows syntax about a quantization step which is common on a per-layer basis.

[0060] When the flag "matrix_adaptive_quant_flag" is 1 (true), the compression

controlling unit 102 determines the quantization steps shown in FIG. 10 or the quantization steps shown in FIG. 12, as quantization steps for weight information of a plurality of respective edges in the network. FIG. 14 shows the syntax of the quantization steps shown in FIG. 10 or 12.

[0061] The syntax shown in FIG. 14 will be described.

As described above, the input layer (first layer) has no edges.

Hence, quantization steps are set for a $\text{layer_id}+1$ th layer in which the ID information "layer_id" is 1 to L-1.

First, in the $\text{layer_id}+1$ th layer, the compression controlling unit 102 sets $\text{base_quant_step}[j]$ ($j = 0, 1, \dots, E-1$) indicating a quantization step for the first node shown in FIG. 10 (the first kernel in FIG. 12).

Note that E is the number of edges $N_{\text{layer_id}-1}$ or the kernel size $K_{\text{layer_id}}$.

[0062] Then, for the second and subsequent nodes (or kernels) ($i \geq 1$), the compression controlling unit 102 refers, on a node-by-node (or kernel-by-kernel) basis, to the flag " $\text{prev_quant_copy_flag}[i-1]$ " indicating whether the quantization step is the same as that for a node (or a kernel) with an immediately previous index.

When the flag " $\text{prev_quant_copy_flag}[i-1]$ " is 1 (true), an $i+1$ th node (or kernel) has the same quantization step as that for an i th node (or kernel).

On the other hand, when the flag " $\text{prev_quant_copy_flag}[i-1]$ " is 0 (false), the compression controlling unit 102 sets the difference value " $\text{diff_quant_value}[i-1][j]$ " ($i = 1, 2, \dots, C-1$, and $j = 0, 1, \dots, E-1$) as information for generating a quantization step for the $i+1$ th node (or kernel).

[0063] The quantization step can be generated by adding a quantization step set for an immediately previous node (kernel) to the difference value " $\text{diff_quant_value}[i-1][j]$ ".

Namely, in the second node (or kernel) ($i = 1$), $\text{base_quant_step}[j] +$

`diff_quant_value[0][j]` is a quantization step. In the third and subsequent nodes (or kernels) ($i \geq 2$), `diff_quant_value[i-2][j] + diff_quant_value[i-1][j]` is a quantization step.

[0064] Note that although the difference value "`diff_quant_value[i-1][j]`" between the quantization steps of nodes (or kernels) is shown as an encoding parameter, an independent quantization step may be set on a per-node (or kernel) basis.

Configuring in this manner improves the encoding efficiency in the encoding unit 103 when there is a tendency that the correlation between the quantization steps of nodes (or kernels) is low.

[0065] The syntax shown in FIG. 15 will be described.

As described above, the input layer (first layer) has no edges.

Hence, the compression controlling unit 102 sets `base_layer_quant_step` as a quantization step which is common to all edge weight information in the second layer.

Then, for the third and subsequent layers ($i \geq 2$), the compression controlling unit 102 refers to the flag "`layer_quant_copy_flag[i-2]`" ($i = 2, 3, \dots, L-1$) indicating whether a quantization step which is common to all edge weight information in an $i+1$ th layer is the same as a quantization step which is common to all edge weight information in an i th layer.

[0066] When the flag "`layer_quant_copy_flag[i-2]`" is 1 (true), the compression controlling unit 102 sets the same quantization step as the quantization step which is common to all edge weight information in the i th layer, as a quantization step which is common to all edge weight information in the $i+1$ th layer. On the other hand, when the flag "`layer_quant_copy_flag[i-2]`" is 0 (false), the compression controlling unit 102 sets `layer_quant_step[i-2]` as a quantization step which is common to all edge weight information in the $i+1$ th layer.

[0067] Note that although a case is shown in which the compression controlling unit

102 defines `layer_quant_step[i-2]` as a quantization step which is independent on a per-layer basis, `layer_quant_step[i-2]` may be defined as a difference value to a quantization step for an immediately previous layer (ith layer). By defining `layer_quant_step[i-2]` as a difference value, many difference values close to 0 occur, thereby being able to improve the encoding efficiency in the encoding unit 103.

[0068] The encoding unit 103 encodes the encoding parameters in FIGS. 13 to 15 as quantization information to generate compressed data.

Note that although the minimum change unit of quantization steps is units of edges, as shown in FIG. 10, the minimum change unit of quantization steps may be units of nodes (in FIG. 12, units of kernels). This has the same meaning as setting $E = 1$ in FIG. 14. In this case, the quantization steps may be independently encoded on a per-node basis (on a per-kernel basis in FIG. 12).

[0069] In addition, the minimum change unit of quantization steps may be units of layers.

This has the same meaning as always executing only `layer_quant_coding()` without the flag "`matrix_adaptive_quant_flag`" when the flag "`layer_adaptive_quant_flag`" = 1 (true) in FIG. 13. By thus making the minimum change unit of quantization steps larger than units of edges, the data size of pre-encoding quantization information can be reduced.

[0070] As described above, in the data processing device 100 according to the first embodiment, the data processing unit 101 processes input data using a neural network. The compression controlling unit 102 determines quantization steps, and generates quantization information that defines the quantization steps. The encoding unit 103 encodes network configuration information including parameter data which is quantized using the quantization steps determined by the compression controlling unit 102, and

the quantization information, to generate compressed data.

Particularly, in the above-description, an example is described in which weight information assigned to edges that connect nodes in a neural network is handled as parameter data of the neural network. By having these configurations, quantization information that defines quantization steps, and network configuration information including parameter data which is quantized using the quantization steps in the quantization information are encoded into compressed data. Accordingly, the parameter data of the neural network can be highly compressed.

In addition, by using quantization information and network configuration information which are decoded from the compressed data, a neural network optimized on the encoding side can be constructed on the decoding side.

[0071] In the data processing device 200 according to the first embodiment, the data processing unit 202 processes input data using a neural network. The decoding unit 201 decodes compressed data.

In this configuration, the data processing unit 202 inversely quantizes parameter data using quantization information and network configuration information which are decoded from compressed data by the decoding unit 201, and constructs a neural network using the network configuration information including the inversely quantized parameter data.

Accordingly, a neural network optimized on the encoding side can be constructed using the quantization information and network configuration information decoded from the compressed data.

[0072] In the data processing device 100 according to the first embodiment, the compression controlling unit 102 changes quantization steps on an edge-by-edge basis. The encoding unit 103 encodes quantization information that defines the edge-by-edge

quantization steps. By such a configuration, parameter data can be quantized with high accuracy.

[0073] In the data processing device 100 according to the first embodiment, the compression controlling unit 102 changes quantization steps on a node-by-node or kernel-by-kernel basis. The encoding unit 103 encodes quantization information that defines the node-by-node or kernel-by-kernel quantization steps.

By such a configuration, too, parameter data can be quantized with high accuracy.

[0074] In the data processing device 100 according to the first embodiment, the compression controlling unit 102 changes quantization steps on a layer-by-layer basis of a neural network. The encoding unit 103 encodes quantization information that defines the layer-by-layer quantization steps for the neural network.

By such a configuration, the compression ratio of parameter data by quantization increases, and thus, the amount of data of pre-encoding weight information can be reduced.

[0075] Second Embodiment.

The first embodiment describes an example in which output results of a neural network are directly used as data processing results, but there is an exemplary application in which outputs of an intermediate layer of a neural network are used as features for data processing on image data and audio data, e.g., image retrieval or matching in the following reference, and the features are subjected to another data processing technique as shown in the following reference, by which final data processing results are obtained.

For example, when outputs of an intermediate layer of a neural network are used as image features for image processing such as image retrieval, matching, or object

tracking, by substituting the outputs for or adding the outputs to image features such as a Histogram of Oriented Gradients (HOG), Scale Invariant Feature Transform (SIFT), and Speeded Up Robust Features (SURF) which are image features used in the above-described conventional image processing, image processing can be implemented using the same processing flow as that of image processing that uses the above-described conventional image features.

In this case, what is to be encoded as network configuration information and quantization information by the data processing device 100 is a portion of the neural network up to the intermediate layer in which outputs serving as features for data processing are obtained.

Furthermore, the data processing device 100 performs data processing such as image retrieval, using the above-described features for data processing. The data processing device 200 decodes the portion of the neural network up to the intermediate layer from compressed data, and performs data processing such as image retrieval, using outputs obtained by inputting input data, as features for the data processing.

(Reference) ISO/IEC JTC1/SC29/WG11/m39219, "Improved retrieval and matching with CNN feature for CDVA", Chengdu, China, Oct. 2016.

[0076] Therefore, in the data processing device 100 according to the second embodiment, the compression ratio of parameter data by quantization increases, thereby being able to reduce the amount of data of pre-encoding weight information. In the data processing device 200 according to the second embodiment, by creating a neural network by decoding compressed data which is outputted from the data processing device 100, data processing can be performed.

[0077] Note that the present invention is not limited to the above-described embodiments, and modifications to any component of the embodiments or omissions of

any component of the embodiments are possible within the scope of the present invention.

INDUSTRIAL APPLICABILITY

[0078] Data processing devices according to the invention can highly compress parameter data of a neural network and thus can be used in, for example, image recognition techniques.

REFERENCE SIGNS LIST

[0079] 10-1 to 10-9, 11-1 to 11-3: node, 12-1 to 12-15: edge, 20: kernel, 100, 200: data processing device, 101, 202: data processing unit, 102: compression controlling unit, 103: encoding unit, 201: decoding unit, 300: processing circuit, 301: processor, 302: memory.

CLAIMS

1. A data processing device comprising:
 - a data processing unit for processing input data using a neural network;
 - a compression controlling unit for determining quantization steps and generating quantization information that defines the quantization steps, the quantization steps being used when parameter data of the neural network is quantized; and
 - an encoding unit for encoding network configuration information and the quantization information to generate compressed data, the network configuration information including the parameter data quantized using the quantization steps determined by the compression controlling unit.
2. A data processing device comprising:
 - a data processing unit for processing input data using a neural network; and
 - a decoding unit for decoding compressed data obtained by encoding quantization information and network configuration information, the quantization information defining quantization steps used when parameter data of the neural network is quantized, and the network configuration information including the parameter data quantized using the quantization steps in the quantization information, wherein
 - the data processing unit inversely quantizes the parameter data using the quantization information and the network configuration information which are decoded from the compressed data by the decoding unit, and constructs the neural network using the network configuration information including the inversely quantized parameter data.
3. The data processing device according to claim 1 or 2, wherein the parameter data of the neural network is weight information assigned to edges that connect nodes in the neural network.
4. The data processing device according to claim 1, wherein
 - the compression controlling unit changes the quantization steps on an edge-by-edge basis, and
 - the encoding unit encodes the quantization information that defines the edge-by-edge quantization steps.

5. The data processing device according to claim 1, wherein
the compression controlling unit changes the quantization steps on a node-by-node or kernel-by-kernel basis, and
the encoding unit encodes the quantization information that defines the node-by-node or kernel-by-kernel quantization steps.
6. The data processing device according to claim 1, wherein
the compression controlling unit changes the quantization steps on a layer-by-layer basis of the neural network, and
the encoding unit encodes the quantization information that defines the layer-by-layer quantization steps for the neural network.
7. A data processing method comprising:
a step of, by a decoding unit, decoding compressed data obtained by encoding quantization information and network configuration information, the quantization information defining quantization steps used when parameter data of a neural network is quantized, the network configuration information including the parameter data quantized using the quantization steps in the quantization information; and
a step of, by a data processing unit, inversely quantizing the parameter data using the quantization information and the network configuration information which are decoded from the compressed data by the decoding unit, constructing the neural network using the network configuration information including the inversely quantized parameter data, and processing input data using the neural network.
8. A data processing method comprising:
a step of, by a data processing unit, processing input data using a neural network;
a step of, by a compression controlling unit, determining quantization steps and generating quantization information that defines the quantization steps, the quantization steps being used when parameter data of the neural network is quantized; and
a step of, by an encoding unit, encoding network configuration information and the quantization information to generate compressed data, the network configuration information including the parameter data quantized using the quantization steps determined by the compression controlling unit.