(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7058801号
(P7058801)

(45)発行日　令和4年4月22日(2022.4.22)

(24)登録日　令和4年4月14日(2022.4.14)

(51)国際特許分類
G06N 3/02 (2006.01)

FI
G06N 3/02

請求項の数　9　（全28頁）

(21)出願番号　特願2021-522553(P2021-522553)
(86)(22)出願日　令和1年9月27日(2019.9.27)
(86)国際出願番号　PCT/JP2019/038133
(87)国際公開番号　WO2021/059476
(87)国際公開日　令和3年4月1日(2021.4.1)
審査請求日　令和3年4月23日(2021.4.23)

(73)特許権者　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目７番３号
(74)代理人　110003166
特許業務法人山王内外特許事務所
(72)発明者　峯澤 彰
東京都千代田区丸の内二丁目７番３号
三菱電機株式会社内
審査官　今城 朋彬

最終頁に続く

(54)【発明の名称】　データ処理装置、データ処理システムおよびデータ処理方法

(57)【特許請求の範囲】
【請求項１】
ニューラルネットワークを学習するデータ処理部と、
前記ニューラルネットワークのモデルを識別するためのモデルヘッダ情報、前記ニューラルネットワークのレイヤを識別するためのレイヤヘッダ情報およびレイヤ単位のエッジの重み情報が符号化された符号化データを生成する符号化部と、
を備え、
前記符号化部は、前記ニューラルネットワークのレイヤ構造を示すレイヤ構造情報と、符号化される各レイヤが参照モデルのレイヤからの更新であるか新規レイヤであるかを示す新規レイヤフラグを符号化すること
を特徴とするデータ処理装置。
【請求項２】
前記符号化部は、レイヤに属するエッジの重み情報を、上位ビットからビットプレーン単位で符号化すること
を特徴とする請求項１記載のデータ処理装置。
【請求項３】
前記符号化部は、前記レイヤヘッダ情報によって識別される、１以上のレイヤに属するエッジの重み情報を符号化すること
を特徴とする請求項１または請求項２記載のデータ処理装置。
【請求項４】

前記符号化部は、エッジの重みの値と特定の値との差分を符号化すること
を特徴とする請求項１または請求項２記載のデータ処理装置。
【請求項５】
前記符号化部は、エッジの重み情報を、ベース符号化データとエンハンスメント符号化データとに分けて符号化すること
を特徴とする請求項１または請求項２記載のデータ処理装置。
【請求項６】
前記符号化部によって生成された符号化データを復号する復号部を備え、
前記データ処理部は、前記復号部によって復号された情報を用いて、前記ニューラルネットワークを学習すること
を特徴とする請求項１または請求項２記載のデータ処理装置。
【請求項７】
ニューラルネットワークを学習する第１のデータ処理部と、
前記ニューラルネットワークのモデルを識別するためのモデルヘッダ情報、前記ニューラルネットワークのレイヤを識別するためのレイヤヘッダ情報およびレイヤ単位のエッジの重み情報が符号化された符号化データを生成する符号化部と、
を有する第１のデータ処理装置と、
前記符号化部によって生成された符号化データから復号する復号部と、
前記復号部によって復号された情報を用いて、前記ニューラルネットワークを生成し、前記ニューラルネットワークを用いたデータ処理を行う第２のデータ処理部と、
を有する第２のデータ処理装置と、
を備え、
前記符号化部は、前記ニューラルネットワークのレイヤ構造を示すレイヤ構造情報と、符号化される各レイヤが参照モデルのレイヤからの更新であるか新規レイヤであるかを示す新規レイヤフラグを符号化すること
を特徴とするデータ処理システム。
【請求項８】
前記符号化部は、前記ニューラルネットワークの中間レイヤまでに関する情報を符号化し、
前記第２のデータ処理装置は、前記ニューラルネットワークの中間レイヤから出力されたデータを特徴量として用いたデータ処理を行うこと
を特徴とする請求項７記載のデータ処理システム。
【請求項９】
データ処理部が、ニューラルネットワークを学習するステップと、
符号化部が、前記ニューラルネットワークのモデルを識別するためのモデルヘッダ情報、前記ニューラルネットワークのレイヤを識別するためのレイヤヘッダ情報およびレイヤ単位のエッジの重み情報が符号化された符号化データを生成するステップと、
を備え、
前記符号化部は、前記ニューラルネットワークのレイヤ構造を示すレイヤ構造情報と、符号化される各レイヤが参照モデルのレイヤからの更新であるか新規レイヤであるかを示す新規レイヤフラグを符号化すること
を特徴とするデータ処理方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ニューラルネットワークの構成に関する情報が符号化された符号化データを生成するデータ処理装置、データ処理システムおよびデータ処理方法に関する。
【背景技術】
【０００２】
入力データの分類（識別）問題および回帰問題を解決する方法として機械学習がある。機械学習には、脳の神経回路（ニューロン）を模擬したニューラルネットワークという手法

がある。ニューラルネットワーク（以下、ＮＮと記載する）では、ニューロンが相互に結合されたネットワークによって表現された確率モデル（識別モデル、生成モデル）によって、入力データの分類（識別）または回帰が行われる。

【０００３】

また、ＮＮは、大量のデータを用いた学習によってＮＮのパラメータを最適化することで、高性能化することができる。ただし、近年のＮＮは大規模化しており、ＮＮのデータサイズが大容量化の傾向にあり、ＮＮを用いたコンピュータの計算負荷も増加している。

【０００４】

例えば、非特許文献１には、ＮＮの構成を示す情報であるエッジの重み（バイアス値を含む）を、スカラ量子化した上で符号化する技術が記載されている。エッジの重みをスカラ量子化した上で符号化することで、エッジに関するデータのデータサイズが圧縮される。

【先行技術文献】

【非特許文献】

【０００５】

【文献】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．

【発明の概要】

【発明が解決しようとする課題】

【０００６】

データ伝送ネットワークを介して、複数のクライアントが、サーバと繋がったシステムにおいて、サーバ側で学習されたＮＮの構造を示すデータを符号化して、符号化データをクライアント側で復号することで、複数のクライアントのそれぞれが、サーバで学習されたＮＮを用いてデータ処理を行うものがある。従来のシステムにおいては、ＮＮの構造を更新する場合、更新されたレイヤに関する情報に加え、更新されなかったレイヤに関する情報についてもクライアントへ伝送される。このため、伝送されるデータサイズの削減ができないという課題があった。

【０００７】

本発明は上記課題を解決するものであって、ＮＮの構造を示すデータのデータサイズを削減することができるデータ処理装置、データ処理システムおよびデータ処理方法を得ることを目的とする。

【課題を解決するための手段】

【０００８】

本発明に係るデータ処理装置は、ＮＮを学習するデータ処理部と、ＮＮのモデルを識別するためのモデルヘッダ情報、ＮＮのレイヤを識別するためのレイヤヘッダ情報およびレイヤ単位のエッジの重み情報が符号化された符号化データを生成する符号化部とを備え、符号化部は、ニューラルネットワークのレイヤ構造を示すレイヤ構造情報と、符号化される各レイヤが参照モデルのレイヤからの更新であるか新規レイヤであるかを示す新規レイヤフラグを符号化する。

【発明の効果】

【０００９】

本発明によれば、符号化部が、ＮＮのレイヤ構造を示すレイヤ構造情報と、符号化される各レイヤが参照モデルのレイヤからの更新であるか新規レイヤであるかを示す新規レイヤフラグを符号化する。ＮＮの構造を示すデータのうち、更新されたレイヤに関する情報のみが符号化されて伝送されるので、ＮＮの構造を示すデータのデータサイズを削減することができる。

【図面の簡単な説明】

【００１０】

【図１】実施の形態１に係るデータ処理システムの構成を示すブロック図である。
【図２】ＮＮの構成例を示す図である。
【図３】実施の形態１に係るデータ処理装置（エンコーダ）の構成を示すブロック図である。
【図４】実施の形態１に係るデータ処理装置（デコーダ）の構成を示すブロック図である。
【図５】実施の形態１に係るデータ処理装置（エンコーダ）の動作を示すフローチャートである。
【図６】実施の形態１に係るデータ処理装置（デコーダ）の動作を示すフローチャートである。
【図７】実施の形態１における符号化データの例を示す図である。
【図８】実施の形態１における符号化データの別の例を示す図である。
【図９】実施の形態１における１次元データの畳み込み処理の例を示す図である。
【図１０】実施の形態１における２次元データの畳み込み処理の例を示す図である。
【図１１】ＮＮの１層目のレイヤにおけるノードごとのエッジの重み情報のマトリクスを示す図である。
【図１２】ＮＮの１層目のレイヤにおけるノードごとのエッジの重み情報の量子化ステップのマトリクスを示す図である。
【図１３】畳み込み層におけるエッジの重み情報のマトリクスを示す図である。
【図１４】畳み込み層におけるエッジの重み情報の量子化ステップのマトリクスを示す図である。
【図１５】実施の形態１に係るデータ処理装置（エンコーダ）の変形例の構成を示すブロック図である。
【図１６】実施の形態１における符号化データの更新の概要を示す図である。
【図１７】図１６に示す符号化データの更新に対応したネットワークモデルの構成を示す図である。
【図１８】モデル情報ヘッダに含まれるレイヤ構造情報の例を示す図である。
【図１９】モデル情報ヘッダに含まれるレイヤ構造情報に対応するレイヤ識別情報の例を示す図である。
【図２０】図２０Ａは、実施の形態１に係るデータ処理装置の機能を実現するハードウェア構成を示すブロック図である。図２０Ｂは、実施の形態１に係るデータ処理装置の機能を実現するソフトウェアを実行するハードウェア構成を示すブロック図である。
【発明を実施するための形態】
【００１１】
実施の形態１．
図１は、実施の形態１に係るデータ処理システムの構成を示すブロック図である。図１に示すデータ処理システムにおいて、サーバ１は、データ伝送ネットワーク２を介して、クライアント３－１，３－２，・・・，３－Ｎと繋がっている。Ｎは、２以上の自然数である。サーバ１は、大量のデータを用いた学習によってＮＮ（ニューラルネットワーク）のパラメータを最適化して、高性能なＮＮを生成するデータ処理装置であり、図１に示すデータ処理システムが備える第１のデータ処理装置である。
【００１２】
データ伝送ネットワーク２は、サーバ１とクライアント３－１，３－２，・・・，３－Ｎとの間でやり取りされるデータが伝送されるネットワークであり、インターネットまたはイントラネットである。例えば、データ伝送ネットワーク２には、ＮＮを生成するための情報が、サーバ１からクライアント３－１，３－２，・・・，３－Ｎへ送信される。
【００１３】
クライアント３－１，３－２，・・・，３－Ｎは、サーバ１によって学習されたＮＮを生成し、生成されたＮＮを用いたデータ処理を行う機器である。例えば、クライアント３－１，３－２，・・・，３－Ｎは、パーソナルコンピュータ（ＰＣ）、カメラ、またはロボットといった、通信機能およびデータ処理機能を有した機器である。クライアント３－１

，３－２，・・・，３－Ｎのそれぞれは、図１に示すデータ処理システムが備える第２のデータ処理装置である。

【００１４】

図１に示すデータ処理システムにおいて、クライアント３－１，３－２，・・・，３－Ｎのそれぞれは、ＮＮのモデルおよびパラメータのデータサイズおよび適切な性能を示す値が異なる。このため、非特許文献１に記載された技術を用いてＮＮのモデルおよびパラメータを符号化しても、クライアント３－１，３－２，・・・，３－Ｎのそれぞれに適したデータサイズに圧縮する必要があり、符号化の処理負荷が高くなる。

【００１５】

そこで、実施の形態１に係るデータ処理システムでは、サーバ１が、ＮＮのモデルを識別するためのモデルヘッダ情報と、ＮＮのレイヤを識別するためのレイヤヘッダ情報と、レイヤ単位のバイアス値を含んだエッジの重み（以降、特に明記しない限り、エッジの重みはバイアス値を含むものとする）の情報とが符号化された符号化データを生成し、データ伝送ネットワーク２を介してクライアント３－１，３－２，・・・，３－Ｎに送信する。クライアント３－１，３－２，・・・，３－Ｎのそれぞれは、データ伝送ネットワーク２を介してサーバ１から伝送された符号化データのうち、必要なレイヤに関する情報のみを復号することが可能である。これにより、サーバ１における符号化の処理負荷が低減され、サーバ１からデータ伝送ネットワーク２へ伝送されるデータサイズを削減することができる。

【００１６】

ここで、ＮＮの構成について説明する。図２は、ＮＮの構成例を示す図である。図２に示すように、入力データ（x_1，x_2，・・・，x_{N1}）は、ＮＮが有するそれぞれの層で処理され、処理結果（y_1，・・・，y_{NL}）が出力される。N_l（ｌ＝１，２，・・・，Ｌ）は、ｌ層目のレイヤのノード数を示しており、Ｌは、ＮＮのレイヤ数を示している。ＮＮは、図２に示すように、入力層、隠れ層および出力層を有しており、これらの層のそれぞれには、複数のノードがエッジで繋がれた構造になっている。複数のノードのそれぞれの出力値は、エッジによって繋がれた前の層のノードの出力値と、エッジの重みおよび層ごとに設定された活性化関数とから算出することができる。

【００１７】

ＮＮには、例えば、全結合層（Ｆｕｌｌｙ－ｃｏｎｎｅｃｔｅｄ　ｌａｙｅｒ）だけでなく、畳み込み層（Ｃｏｎｖｏｌｕｔｉｏｎａｌ　ｌａｙｅｒ）とプーリング層（Ｐｏｏｌｉｎｇ　ｌａｙｅｒ）とを有する畳み込みＮＮ（ＣＮＮ；Ｃｏｎｖｏｌｕｔｉｏｎａｌ　Ｎｅｕｒａｌ　Ｎｅｔｗｏｒｋ）がある。ＣＮＮでは、データのフィルタ処理を実現するネットワークなど、分類および回帰以外のデータ処理を実現するネットワークが生成可能である。

【００１８】

例えば、画像または音声を入力として、入力信号のノイズ除去または高品質化を実現する画像または音声のフィルタ処理、圧縮音声の高域が失われた音声の高域復元処理、一部の画像領域が欠損した画像の復元処理（ｉｎｐａｉｎｔｉｎｇ）、あるいは画像の超解像処理がＣＮＮで実現可能である。ＣＮＮでは、生成モデルによって生成されたデータであるか否かを判定する識別モデルを用いてデータの真贋を判定する、生成モデルと識別モデルを組み合わせたＮＮを構築することもできる。

【００１９】

近年では、生成モデルが、識別モデルによって真のデータでないと見破られるデータを生成しないように、識別モデルが、生成モデルによって生成されたデータを真のデータでないと見破れるように敵対的に学習された敵対的生成ネットワーク（Ｇｅｎｅｒａｔｉｖｅ　Ａｄｖｅｒｓａｒｉａｌ　Ｎｅｔｗｏｒｋ）という新しいＮＮも提案されている。このＮＮでは、高精度な生成モデルおよび識別モデルを生成することが可能である。

【００２０】

図３は、実施の形態１に係るデータ処理装置（エンコーダ）の構成を示すブロック図であ

る。図３に示すデータ処理装置は、学習用データセットと評価用データセットを用いてＮＮを学習し、ＮＮの構成を示すモデル情報（以下、モデル情報と記載する）の符号化データを生成する第１のデータ処理装置であり、例えば、図１に示したサーバ１である。

【００２１】

図３に示すデータ処理装置は、データ処理部１０および符号化部１１を備える。データ処理部１０は、ＮＮを学習する第１のデータ処理部であり、学習部１０１、評価部１０２および制御部１０３を備える。符号化部１１は、学習部１０１によって学習されたＮＮのモデルを識別するモデルヘッダ情報、ＮＮのレイヤを識別するレイヤヘッダ情報、およびレイヤ単位のエッジの重み情報が符号化された符号化データを生成する。また、符号化部１１は、符号化するレイヤ（符号化レイヤ）のレイヤ構造情報を符号化し、新規レイヤフラグを符号化する。レイヤ構造情報は、ＮＮのレイヤ構造を示す情報である。新規レイヤフラグは、当該レイヤが新たに追加されるレイヤであるか既にあるレイヤを更新したレイヤであるかを識別するためのフラグ情報であり、詳細は後述する。

【００２２】

学習部１０１は、学習用データセットを用いてＮＮの学習処理を実施し、学習したＮＮのモデル情報を生成する。モデル情報は、学習部１０１から評価部１０２に出力される。さらに、学習部１０１は、後述する制御部１０３によって制御された符号化用モデル情報を持ち、制御部１０３によって学習完了指示を受けた場合に符号化用モデル情報を符号化部１１に出力する。評価部１０２は、モデル情報を用いてＮＮを生成し、生成されたＮＮを用いて、評価用データセットから推論処理を実施する。推論処理の結果として得られた評価指標の値が評価結果であり、評価結果は、評価部１０２から、制御部１０３に出力される。評価指標は、評価部１０２に設定されており、例えば、推論精度または損失関数の出力値である。

【００２３】

制御部１０３は、評価部１０２によって評価結果として得られた評価値から、学習部１０１によって学習されたＮＮのモデルの更新の有無と、学習部１０１によるＮＮの学習の完了可否とを判定し、判定結果に応じて学習部１０１を制御する。例えば、制御部１０３は、評価値をモデル更新判定基準と比較して、この比較結果に基づいて当該モデル情報を符号化用モデル情報として更新するか否かを判定する。また、制御部１０３は、評価値を学習完了判定基準と比較して、この比較結果に基づいて学習部１０１によるＮＮの学習を完了するか否かを判定する。なお、これらの判定基準は、評価値の履歴から決定される。

【００２４】

図４は、実施の形態１に係るデータ処理装置（デコーダ）の構成を示すブロック図である。図４に示すデータ処理装置は、図３に示した符号化部１１によって生成された符号化データを復号してＮＮを生成し、生成されたＮＮを用いて、１以上の評価用データを推論処理する第２のデータ処理装置であり、例えば、図１に示したクライアント３－１，３－２，・・・，３－Ｎである。

【００２５】

図４に示すデータ処理装置は、復号部２０１および推論部２０２を備えている。復号部２０１は、符号化部１１によって生成された符号化データから、モデル情報を復号する。例えば、復号部２０１は、図４に示すデータ処理装置において必要な情報のみを、符号化データから復号することができる。

【００２６】

推論部２０２は、復号部２０１によって復号されたモデル情報を用いてＮＮを生成し、生成されたＮＮを用いたデータ処理を実施する第２のデータ処理部である。例えば、データ処理は、ＮＮを用いた評価用データに対する推論処理である。推論部２０２は、ＮＮを用いて評価用データに対する推論処理を実施し、推論結果を出力する。

【００２７】

次に、実施の形態１に係るデータ処理システムの動作について説明する。図５は、実施の形態１に係るデータ処理装置（エンコーダ）の動作を示すフローチャートであり、図３に

示したデータ処理装置によるデータ処理方法を示している。学習部１０１がＮＮを学習する（ステップＳＴ１）。例えば、学習部１０１は、学習用データセットを用いてＮＮの学習を実施し、この学習によって得られたモデル情報を評価部１０２に出力する。

【００２８】

モデル情報は、ＮＮのモデルの構成を示す情報であり、レイヤごとの構造を示すレイヤ構造情報と、レイヤに属する各エッジの重み情報を含んで構成される。レイヤ構造情報には、レイヤ種別情報、レイヤ種別に関わる構成情報、およびエッジの重み以外でレイヤを構成するために必要な情報が含まれている。エッジの重み以外でレイヤを構成するために必要な情報には、例えば、活性化関数がある。レイヤ種別情報は、レイヤの種別を示す情報であり、レイヤ種別情報を参照することで、畳み込み層、プーリング層または全結合層といったレイヤの種別を識別することが可能である。

【００２９】

レイヤ種別に関わる構成情報は、レイヤ種別情報に対応する種別のレイヤの構成を示す情報である。例えば、レイヤ種別情報に対応するレイヤの種別が畳み込み層である場合、レイヤ種別に関わる構成情報には、畳み込みを行うチャネル数、畳み込みフィルタ（カーネル）のデータサイズと形状、畳み込み間隔（ストライド）、畳み込み処理の入力信号の境界に対するパディングの有無、および、パディング有りの場合はパディングの方法がある。また、レイヤ種別情報に対応するレイヤの種別がプーリング層である場合、レイヤ種別に関わる構成情報には、最大プーリングまたは平均プーリングといったプーリング方法、プーリング処理を行うカーネルの形状、プーリング間隔（ストライド）、プーリング処理の入力信号の境界に対するパディングの有無、および、パディング有りの場合はパディングの方法がある。

【００３０】

各エッジの重みを示す情報には、全結合層のように各エッジで独立に重みが設定される場合がある。一方、畳み込み層のように、エッジの重みが畳み込みフィルタ（カーネル）単位（チャネル単位）で共通する、すなわち、一つのフィルタでエッジの重みが共通する場合もある。

【００３１】

評価部１０２がＮＮを評価する（ステップＳＴ２）。例えば、評価部１０２は、学習部１０１によって生成されたモデル情報を用いてＮＮを生成し、生成されたＮＮを用いて、評価用データセットから推論処理を実施する。評価結果は、評価部１０２から制御部１０３に出力される。評価結果は、例えば、推論精度または損失関数の出力値である。

【００３２】

次に、制御部１０３が、モデル情報を更新するか否かを判定する（ステップＳＴ３）。例えば、制御部１０３は、評価部１０２によって生成された評価値がモデル更新判定基準を満たさない場合、学習部１０１が持つ符号化用モデル情報を更新しないと判定し、評価値がモデル更新判定基準を満たす場合、上記符号化用モデル情報を更新すると判定する。

【００３３】

モデル更新判定基準の一例としては、評価値が損失関数の出力値である場合、学習開始時からの学習履歴における評価値の最小値よりも今回の学習による評価値が小さいこと、がある。他の一例としては、評価値が推論精度である場合、学習開始時からの学習履歴における評価値の最大値よりも今回の学習の評価値が大きいこと、がある。

【００３４】

また、学習履歴の切り替え単位も任意としてもよい。例えば、後述するモデル識別番号（ｍｏｄｅｌ＿ｉｄ）ごとに学習履歴を持つとする。この場合、当該モデルが、後述する参照モデル識別番号（ｒｅｆｅｒｅｎｃｅ＿ｍｏｄｅｌ＿ｉｄ）を持たない場合は、学習履歴なしとして学習を開始する。すなわち、１回目のステップＳＴ３では、必ずモデル情報を更新することになる。一方、当該モデルが参照モデル識別番号を持つ場合には、当該参照モデル識別番号が指し示すモデルの学習履歴（履歴Ａ）を参照する。これにより、当該モデルの学習時に参照モデル識別番号が指し示すモデルより評価値が悪い（推論精度が低

い、損失関数の値が大きい、など）モデルに更新されてしまうことを防ぐことが可能となる。このとき、当該モデルのモデル識別番号と参照モデル識別番号が同一である場合、当該モデルの学習を実施する度に、参照モデル識別番号に対応する学習履歴（履歴Ａ）が更新されていくことになる。一方、当該モデルのモデル識別番号と参照モデル識別番号が異なる場合は、参照モデル識別番号に対応する学習履歴（履歴Ａ）を当該モデルのモデル識別番号の学習履歴（履歴Ｂ）の初期値としてコピーした上で、当該モデルの学習を実施する度に当該モデルの学習履歴（履歴Ｂ）が更新されていくことになる。

【００３５】

制御部１０３によってモデル情報を更新すると判定された場合（ステップＳＴ３；ＹＥＳ）、学習部１０１は、符号化用モデル情報を当該モデル情報に更新する（ステップＳＴ４）。例えば、制御部１０３は、モデル情報の更新があることを示すモデル更新指示情報を生成し、モデル更新指示情報を含んだ学習制御情報を学習部１０１に出力する。学習部１０１は、学習制御情報に含まれるモデル更新指示情報に従って、符号化用モデル情報を当該モデル情報に更新する。

【００３６】

一方、モデル情報を更新しないと判定した場合（ステップＳＴ３；ＮＯ）、制御部１０３は、モデル情報の更新がないことを示すモデル更新指示情報を生成し、モデル更新指示情報を含んだ学習制御情報を学習部１０１に出力する。学習部１０１は、学習制御情報に含まれるモデル更新指示情報に従って符号化用モデル情報を更新しない。

【００３７】

次に、制御部１０３は、評価値を学習完了判定基準と比較し、この比較結果に基づいて学習部１０１によるＮＮの学習を完了するか否かを判定する（ステップＳＴ５）。例えば、学習完了判定基準が評価部１０２によって生成された評価値が特定の値に達したか否かとする場合、制御部１０３は、評価部１０２によって生成された評価値が学習完了判定基準を満たすと、学習部１０１によるＮＮの学習が完了したと判定し、評価値が学習完了判定基準を満たしていなければ、学習部１０１によるＮＮの学習が完了していないと判定する。あるいは、例えば、連続でＭ回（Ｍは１以上の予め定められた整数）、モデル情報の更新なし（ステップＳＴ３；ＮＯ）が選択されると、学習完了と判定するなどの、学習完了判定基準が直近の学習履歴に基づくものとする場合、制御部１０３は、学習履歴が学習完了判定基準を満たしていなければ、学習部１０１によるＮＮの学習が完了していないと判定する。

【００３８】

制御部１０３によってＮＮの学習が完了したと判定された場合（ステップＳＴ５；ＹＥＳ）、学習部１０１が符号化用モデル情報を符号化部１１に出力するとともに、ステップＳＴ６の処理に移行する。一方、制御部１０３によってＮＮの学習が完了していないと判定された場合（ステップＳＴ５；ＮＯ）、ステップＳＴ１からの処理が実行される。

【００３９】

符号化部１１は、学習部１０１から入力された符号化用モデル情報を符号化する（ステップＳＴ６）。符号化部１１は、学習部１０１によって生成された符号化用モデル情報を、ＮＮのレイヤ単位で符号化し、ヘッダ情報とレイヤ単位の符号化データから構成された符号化データを生成する。また、符号化部１１は、レイヤ構造情報を符号化し、新規レイヤフラグを符号化する。

【００４０】

図６は、実施の形態１に係るデータ処理装置（デコーダ）の動作を示すフローチャートであり、図４に示したデータ処理装置の動作を示している。復号部２０１は、符号化部１１によって符号化された符号化データからモデル情報を復号する（ステップＳＴ１１）。次に、推論部２０２は、復号部２０１によって復号されたモデル情報からＮＮを生成する（ステップＳＴ１２）。推論部２０２は、生成されたＮＮを用いて、評価用データに対する推論処理を実施し、推論結果を出力する（ステップＳＴ１３）。

【００４１】

次に、図５のステップＳＴ６における符号化部１１によるモデル情報の符号化について詳細に説明する。符号化部１１によるモデル情報の号化には、例えば（１）または（２）の符号化方法を用いることができる。あるいは、（１）または（２）の符号化がパラメータ毎にどちらを用いるか定義されていてもよい。例えば、ヘッダ情報は（１）、重みデータは（２）とすることで、デコーダは、ヘッダ情報を可変長復号することなく容易に解析可能としつつ、符号化データのデータサイズの大部分を占める重みデータは可変長復号によって高い圧縮を実現することができ、符号化データ全体のデータサイズを抑えることができる。

（１）モデル情報に含まれる各情報を構成するパラメータが、パラメータに定義されているビット精度で記述されたビット列そのものが、ヘッダ情報が存在する場合はヘッダ情報を含めて予め設定された順序で並べられたデータを符号化データとする。ビット精度は、例えば、ｉｎｔ型８ビットあるいはｆｌｏａｔ型３２ビットといった、パラメータに定義されているビット精度である。

（２）モデル情報に含まれる各情報を構成するパラメータが、パラメータごとに設定された可変長符号化方法によって符号化されたビット列そのものが、ヘッダ情報を含めて予め設定された順序で並べられたデータを符号化データとする。

【００４２】

図７は、実施の形態１における符号化データの例を示す図であり、上記（１）または（２）の符号化データは、図７に示す順序で並べてもよい。図７に示す符号化データは、データユニットと呼ぶデータの集まりから構成され、データユニットには、非レイヤデータユニットとレイヤデータユニットがある。レイヤデータユニットは、レイヤ単位の符号化データであるレイヤデータが格納されるデータユニットである。

【００４３】

レイヤデータは、スタートコード、データユニットタイプ、レイヤ情報ヘッダ、および重みデータから構成される。レイヤ情報ヘッダは、ＮＮのレイヤを識別するためのレイヤヘッダ情報が符号化されたものである。重みデータは、レイヤ情報ヘッダが示すレイヤに属するエッジの重み情報が符号化されたものである。なお、図７に示す符号化データにおいて、各レイヤデータユニットの並び順は必ずしもＮＮの各層の並び順と同じでなくてもよく、任意である。これは、後述するレイヤ識別番号（ｌａｙｅｒ＿ｉｄ）によって、各レイヤデータユニットがＮＮのどの位置のレイヤであるかを識別可能であるからである。

【００４４】

非レイヤデータユニットは、レイヤデータ以外のデータが格納されるデータユニットである。例えば、非レイヤデータユニットには、スタートコード、データユニットタイプ、およびモデル情報ヘッダが格納されている。モデル情報ヘッダは、ＮＮのモデルを識別するためのモデルヘッダ情報が符号化されたものである。

【００４５】

スタートコードは、データユニットの先頭位置に格納され、データユニットの先頭位置を識別するためのコードである。クライアント３－１，３－２，・・・，３－Ｎ（以下、復号側と記載する）は、スタートコードを参照することにより、非レイヤデータユニットまたはレイヤデータユニットの先頭位置を特定することが可能である。例えば、スタートコードとして０ｘ０００００１が定義された場合、データユニットに格納されたスタートコード以外のデータは、０ｘ０００００１が発生しないように設定される。これにより、スタートコードからデータユニットの先頭位置を特定することができる。

【００４６】

０ｘ０００００１が発生しないように設定するためには、例えば、０ｘ０００００００～０ｘ０００００３の符号化データにおける３バイト目に０３を挿入して０ｘ０００３００～０ｘ０００３０３とし、復号するときに、０ｘ０００３を０ｘ０００００と変換することにより、元に戻すことができる。なお、スタートコードは、一意に識別可能なビット列であれば、０ｘ０００００１以外のビット列をスタートコードとして定義してもよい。また、データユニットの先頭位置を識別可能な方法であれば、スタートコードを用いなくてもよ

い。例えば、データユニットの終端であることを識別可能なビット列をデータユニットの終端に付けてもよい。あるいは、非レイヤデータユニットの先頭のみスタートコードを付けることとし、モデル情報ヘッダの一部として、各レイヤデータユニットのデータサイズを符号化するようにしてもよい。このようにすることで、上記情報から、各レイヤデータユニットの区切り位置を識別することが可能である。

【００４７】

データユニットタイプは、データユニットにおいてスタートコードの次に格納されて、データユニットの種類を識別するためのデータである。データユニットタイプは、データユニットの種類ごとに予め値が定義されている。復号側は、データユニットに格納されたデータユニットタイプを参照することで、データユニットが、非レイヤデータユニットであるのか、レイヤデータユニットであるのかを識別でき、さらに、どのような非レイヤデータユニットまたはレイヤデータユニットであるのかを識別することが可能である。

【００４８】

非レイヤデータユニットにおけるモデル情報ヘッダには、モデル識別番号（ｍｏｄｅｌ＿ｉｄ）、モデル内レイヤデータユニット数（ｎｕｍ＿ｌａｙｅｒｓ）および符号化レイヤデータユニット数（ｎｕｍ＿ｃｏｄｅｄ＿ｌａｙｅｒｓ）が含まれる。モデル識別番号は、ＮＮのモデルを識別するための番号である。従って、基本的には、個々のモデルにおいて互いに独立した番号を持つが、もし実施の形態１に係るデータ処理装置（デコーダ）が過去に受信したモデルと同一のモデル識別番号を持つモデルを新たに受信した場合は、当該モデル識別番号を持つモデルが上書きされることになる。モデル内レイヤデータユニット数は、モデル識別番号で識別されるモデルを構成するレイヤデータユニットの数である。符号化レイヤデータユニット数は、符号化データの中に、実際に存在するレイヤデータユニットの数である。図７の例では、レイヤデータユニット（１）～（ｎ）が存在することから、符号化レイヤデータユニット数はｎである。なお、符号化レイヤデータユニット数は、必ず、モデル内レイヤデータユニット数以下になる。

【００４９】

レイヤデータユニットにおけるレイヤ情報ヘッダには、レイヤ識別番号（ｌａｙｅｒ＿ｉｄ）およびレイヤ構造情報が含まれる。レイヤ識別番号は、レイヤを識別するための番号である。レイヤ識別番号によってどの層のレイヤか識別できるように、レイヤ識別番号の値の振り方は予め固定的に定義される。例えば、ＮＮの入力層を０、次の層を１というように、入力層に近い層から順に番号を振る、などである。レイヤ構造情報は、ＮＮのレイヤごとの構成を示す情報であって、レイヤ種別情報、レイヤ種別に関わる構成情報、およびエッジの重み以外にレイヤを構成するために必要な情報を含んでいる。例えば、後述するｍｏｄｅｌ＿ｓｔｒｕｃｔｕｒｅ＿ｉｎｆｏｒｍａｔｉｏｎとｌａｙｅｒ＿ｉｄ＿ｉｎｆｏｒｍａｔｉｏｎの当該レイヤ部分のみの情報である。さらに、レイヤ構造情報として当該レイヤの各エッジの重みのビット精度を示すｗｅｉｇｈｔ＿ｂｉｔ＿ｌｅｎｇｔｈを持つ。例えば、ｗｅｉｇｈｔ＿ｂｉｔ＿ｌｅｎｇｔｈ＝８であれば、重みは８ビットのデータであることを示す。したがって、レイヤ単位にエッジの重みのビット精度を設定することができる。これによって、レイヤの重要度（ビット精度が出力結果に影響する程度）に応じてレイヤ単位にビット精度を変更する等の適応制御が可能となる。

【００５０】

なお、これまでレイヤ構造情報を含むレイヤ情報ヘッダを示したが、モデル情報ヘッダが、符号化データに含まれる全てのレイヤ構造情報（ｍｏｄｅｌ＿ｓｔｒｕｃｔｕｒｅ＿ｉｎｆｏｒｍａｔｉｏｎ）と本レイヤ構造情報に対応するレイヤ識別情報（ｌａｙｅｒ＿ｉｄ＿ｉｎｆｏｒｍａｔｉｏｎ）とを含んでもよい。復号側は、モデル情報ヘッダを参照することで、各レイヤ識別番号のレイヤの構成を特定することができる。さらに、上記の場合はモデル情報ヘッダを参照することで、各レイヤ識別番号のレイヤの構成を特定することができるため、レイヤ情報ヘッダは、レイヤ識別番号のみを持つようにしてもよい。このようにすることで、レイヤデータユニットのデータサイズが非レイヤデータユニットのデータサイズよりも大きい場合、各レイヤデータユニットのデータサイズを小さくするこ

とができ、符号化データ内のデータユニットの最大データサイズを小さくすることができる。

【００５１】

レイヤデータユニットにおいて、レイヤ情報ヘッダの次に、レイヤ単位に符号化された重みデータが格納されている。重みデータは、非零フラグおよび非零重みデータを含んでいる。非零フラグは、エッジの重みの値が零か否かを示すフラグであり、対応するレイヤに属する全てのエッジの重みについての非零フラグが設定される。

【００５２】

非零重みデータは、重みデータにおいて非零フラグに続いて設定されるデータであり、非零フラグが非零（有意）を示す重みについて、その重みの値が設定されたものである。図７において、それぞれが非零の重みの値を示す重みデータ（１）～重みデータ（ｍ）が非零重みデータとして設定されている。非零の重みデータ数ｍは、対応するレイヤｌの全ての重みの数Ｍｌ以下である。なお、重みの値が非零のエッジが疎であるレイヤに関する重みデータは、非零重みデータが少なく、ほぼ非零フラグのみとなるため、重みデータのデータサイズが大きく削減される。

【００５３】

図８は、実施の形態１における符号化データの別の例を示す図であり、上記（１）または（２）の符号化データは図８に示す順序で並べてもよい。図８に示す符号化データは、重みデータのデータ構成が図７と異なっており、非零重みデータには、対応するレイヤに属する全てのエッジの重みが上位ビットから順にビットプレーンごとにまとめて並べられている。さらに、レイヤ情報ヘッダには、エッジの重みを示す各ビットの先頭位置を示すビットプレーンデータ位置識別情報が設定されている。

【００５４】

例えば、エッジの重みに定義されたビット精度がＸであると、対応するレイヤに属する全てのエッジの重みは、ビット精度Ｘでそれぞれ記述される。符号化部１１は、これらの重みのビット列のうち、１ビット目の非零重みデータである、１ビット目の重みデータ（１），１ビット目の重みデータ（２），・・・，１ビット目の重みデータ（ｍ）を、１ビット目の各非零重みデータに設定する。この処理は、２ビット目の非零重みデータからＸビット目の非零重みデータまで繰り返される。なお、１ビット目の重みデータ（１），１ビット目の重みデータ（２），・・・，１ビット目の重みデータ（ｍ）は、１ビット目のビットプレーンを構成する非零の重みデータである。

【００５５】

復号側は、ビットプレーンデータ位置識別情報に基づいて、レイヤ単位の符号化データのうち、必要な符号化データを特定し、特定された符号化データを任意のビット精度で復号することができる。すなわち、復号側は、符号化データから必要な符号化データのみを選別でき、復号側の環境に応じたＮＮのモデル情報を復号することが可能である。なお、ビットプレーンデータ位置識別情報は、ビットプレーンデータ間の区切り位置を識別可能な情報であればよく、各ビットプレーンデータの先頭位置を示す情報であってもよいし、各ビットプレーンデータのデータサイズを示す情報であってもよい。

【００５６】

ＮＮの構成を示す全ての符号化データを復号側へ伝送するためには、データ伝送ネットワーク２の伝送帯域が十分でない場合に、符号化部１１が、当該符号化データのうち、データ伝送ネットワーク２の伝送帯域に応じて伝送する非零重みデータを制限してもよい。例えば、３２ビット精度で記述された重み情報のビット列のうち、上位８ビットの非零重みデータを伝送対象とする。復号側は、この非零重みデータの次に並ぶスタートコードから、符号化データにおいて、８ビット目の非零重みデータの後に、次のレイヤに対応するレイヤデータユニットが並んでいることを認識できる。また、復号側は、重みデータにおける非零フラグを参照することで、値が零の重みを正しく復号することができる。

【００５７】

復号側で任意のビット精度で重みデータが復号されたときに、そのビット精度での推論精

度を改善するため、符号化部１１は、各ビット精度で復号されたときの重みに加算するオフセットをレイヤ情報ヘッダに含めてもよい。例えば、符号化部１１は、ビット精度で記述された重みのビット列に対してレイヤ単位に一様なオフセットを加算し、最も高精度になるオフセットを求めて、求められたオフセットを、レイヤ情報ヘッダに含めて符号化する。

【００５８】

また、符号化部１１は、ＮＮが備える全てのレイヤにおけるエッジの重みのオフセットをモデル情報ヘッダに含めて符号化してもよい。さらに、符号化部１１は、オフセットを含むか否かを示すフラグをレイヤ情報ヘッダまたはモデル情報ヘッダに設定し、例えば、フラグが有効である場合のみ、オフセットを符号化データに含めてもよい。

【００５９】

符号化部１１は、エッジの重みの値と特定の値の差分を符号化対象としてもよい。特定の値としては、例えば、符号化順が一つ前の重みが挙げられる。また、一つ上位のレイヤ（入力層に近いレイヤ）に属する、対応するエッジの重みを特定の値としてもよいし、更新前のモデルの対応するエッジの重みを特定の値としてもよい。

【００６０】

さらに、符号化部１１は、（Ａ）、（Ｂ）および（Ｃ）に示す機能を有する。

（Ａ）符号化部１１は、ベース符号化データとエンハンスメント符号化データとに分けて符号化するスケーラブル符号化機能を有する。

（Ｂ）符号化部１１は、基準のＮＮにおけるエッジの重みとの差分を符号化する機能を有する。

（Ｃ）符号化部１１は、基準のＮＮにおける部分的な情報（例えば、レイヤ単位の情報）のみを、ＮＮの更新用情報として符号化する機能を有する。

【００６１】

（Ａ）の例について説明する。

符号化部１１は、エッジの重みについて予め定義された量子化手法を用いて、エッジの重みを量子化し、量子化後の重みを符号化したデータをベース符号化データとし、量子化誤差を重みとみなして符号化したデータをエンハンスメント符号化データとする。ベース符号化データとされた重みは、量子化によって量子化前の重みよりもビット精度が低下するため、データサイズが削減される。復号側へ符号化データを伝送する伝送帯域が十分でない場合に、実施の形態１に係るデータ処理装置は、ベース符号化データのみを復号側に伝送する。一方、復号側へ符号化データを伝送する伝送帯域が十分な場合、実施の形態１に係るデータ処理装置は、ベース符号化データに加え、エンハンスメント符号化データも含めて復号側に伝送する。

【００６２】

エンハンスメント符号化データは２つ以上とすることができる。例えば、符号化部１１は、量子化誤差をさらに量子化したときの量子化値を、一つ目のエンハンスメント符号化データとし、その量子化誤差を２つ目のエンハンスメント符号化データとする。さらに、２つ目のエンハンスメント符号化データの量子化誤差をさらに量子化した量子化値とその量子化誤差とに分けて目的のエンハンスメント符号化データの数になるように符号化してもよい。このように、スケーラブル符号化を用いることで、データ伝送ネットワーク２の伝送帯域と伝送許容時間とに応じた符号化データの伝送が可能である。

【００６３】

なお、符号化部１１は、図８に示した非零重みデータの上位Ｍビットまでをベース符号化データとして符号化し、残りのビット列を１以上に分割して１以上のエンハンスメント符号化データとしてもよい。この場合、符号化部１１は、ベース符号化データとエンハンスメント符号化データのそれぞれで非零フラグを再び設定する。上位ビットのエンハンスメント符号化データにおいて０となった重みは、必ず０となる。

【００６４】

（Ｂ）の例について説明する。

符号化部１１は、学習部１０１による再学習前のＮＮのモデルが存在する場合、再学習後のＮＮのモデルにおけるエッジの重みと、再学習前のモデルにおける対応するエッジの重みとの差分を符号化してもよい。なお、再学習には、転移学習または追加学習がある。データ処理システムにおいて、高い頻度でＮＮの構成を更新するか、あるいは再学習ごとの学習データの分布の変化が小さい場合、エッジの重みの差分が小さいので、再学習後の符号化データのデータサイズが削減される。

【００６５】

符号化部１１は、モデル識別番号に加え、参照すべき更新前のモデルを識別するための参照モデル識別番号（ｒｅｆｅｒｅｎｃｅ＿ｍｏｄｅｌ＿ｉｄ）をモデル情報ヘッダに含む。（Ｂ）の例において、上記参照モデル識別番号から再学習前のモデルを識別することが可能となる。さらに、符号化部１１は、符号化データに参照元があるか否かを示すフラグ（ｒｅｆｅｒｅｎｃｅ＿ｍｏｄｅｌ＿ｐｒｅｓｅｎｔ＿ｆｌａｇ）を、モデル情報ヘッダに設定してもよい。このとき、符号化部１１は、まず上記フラグ（ｒｅｆｅｒｅｎｃｅ＿ｍｏｄｅｌ＿ｐｒｅｓｅｎｔ＿ｆｌａｇ）を符号化し、上記フラグがモデルの更新用の符号化データであることを示す場合にのみ、さらにモデル情報ヘッダに参照モデル識別番号を設定する。

【００６６】

例えば、図１に示したデータ処理システムにおいて、クライアント間でＮＮの更新頻度が異なるか、互いに異なるモデルのＮＮを用いてデータ処理を実施する場合であっても、クライアントは、参照モデル識別番号を参照することで、どのモデルに対する更新用の符号化データであるのかを正しく識別することができる。参照モデル識別番号からクライアント側にないモデルの更新用の符号化データであることが識別された場合には、クライアントが、そのことをサーバ１に伝えることも可能である。

【００６７】

（Ｃ）の例について説明する。

学習部１０１は、再学習前のＮＮのモデルが存在する場合、例えばＦｉｎｅ－ｔｕｎｉｎｇを目的として、ＮＮの上位（入力層側）から１以上の任意のレイヤを固定し、一部のレイヤのみを再学習することがある。この場合、符号化部１１は、再学習によって更新されたレイヤの構成を示す情報のみを符号化する。これにより、ＮＮの更新において、復号側へ伝送される符号化データのデータサイズが削減される。なお、符号化データにおける符号化レイヤデータユニット数（ｎｕｍ＿ｃｏｄｅｄ＿ｌａｙｅｒｓ）は、モデル内レイヤデータユニット数（ｎｕｍ＿ｌａｙｅｒｓ）以下となる。復号側では、モデル情報ヘッダに含まれる参照モデル識別番号と、レイヤ情報ヘッダに含まれるレイヤ識別番号とを参照することで、更新すべきレイヤを特定できる。

【００６８】

次に、学習部１０１、評価部１０２および推論部２０２によるデータ処理を説明する。

図９は、実施の形態１における１次元データの畳み込み処理の例を示す図であり、１次元データの畳み込み処理を行う畳み込み層を示している。１次元データには、例えば、音声データ、時系列データがある。図９に示す畳み込み層は、前層に９つのノード１０－１～１０－９、次層に３つのノード１１－１～１１－３を備えている。エッジ１２－１，１２－６，１２－１１には同じ重みが付与されており、エッジ１２－２，１２－７，１２－１２には同じ重みが付与されており、エッジ１２－３，１２－８，１２－１３には同じ重みが付与されており、エッジ１２－４，１２－９，１２－１４には同じ重みが付与されており、エッジ１２－５，１２－１０，１２－１５には同じ重みが付与されている。また、エッジ１２－１から１２－５までの重みは全て異なる値となる場合もあるし、複数の重みが同じ値となる場合もある。

【００６９】

前層の９つのノード１０－１～１０－９のうち、５つのノードが、上記の重みで次層の１つのノードに繋がっている。カーネルサイズＫは５であり、カーネルは、これらの重みの組み合わせによって規定される。例えば、図９に示すように、ノード１０－１は、エッジ

１２－１を介してノード１１－１に繋がり、ノード１０－２は、エッジ１２－２を介してノード１１－１に繋がり、ノード１０－３は、エッジ１２－３を介してノード１１－１に繋がり、ノード１０－４は、エッジ１２－４を介してノード１１－１に繋がり、ノード１０－５は、エッジ１２－５を介してノード１１－１に繋がっている。カーネルは、エッジ１２－１～１２－５の重みの組み合わせによって規定される。

【００７０】

ノード１０－３は、エッジ１２－６を介してノード１１－２に繋がり、ノード１０－４は、エッジ１２－７を介してノード１１－２に繋がり、ノード１０－５は、エッジ１２－８を介してノード１１－２に繋がり、ノード１０－６は、エッジ１２－９を介してノード１１－２に繋がり、ノード１０－７は、エッジ１２－１０を介してノード１１－２に繋がっている。カーネルは、エッジ１２－６～１２－１０の重みの組み合わせによって規定される。

【００７１】

ノード１０－５は、エッジ１２－１１を介してノード１１－３に繋がり、ノード１０－６は、エッジ１２－１２を介してノード１１－３に繋がり、ノード１０－７は、エッジ１２－１３を介してノード１１－３に繋がり、ノード１０－８は、エッジ１２－１４を介してノード１１－３に繋がり、ノード１０－９は、エッジ１２－１５を介してノード１１－３に繋がっている。カーネルは、エッジ１２－１１～１２－１５の重みの組み合わせによって規定される。

【００７２】

学習部１０１、評価部１０２および推論部２０２は、ＣＮＮを用いた入力データの処理において、畳み込み層のエッジの重みの組み合わせを用いて、カーネルごとにステップ数の間隔（図９では、Ｓ＝２）で畳み込み演算を実施する。エッジの重みの組み合わせは、カーネルごとに学習によって決定される。なお、画像認識用途のＣＮＮでは、複数のカーネルを有する畳み込み層でＮＮが構成される場合が多い。

【００７３】

図１０は、実施の形態１における２次元データの畳み込み処理の例を示す図であり、画像データといった２次元データの畳み込み処理を示している。図１０に示す２次元データのうち、カーネル２０は、ｘ方向のサイズがK_x、ｙ方向のサイズがK_yのブロック領域である。カーネルサイズＫは、$K = K_x \times K_y$である。学習部１０１、評価部１０２または推論部２０２は、２次元データにおいて、ｘ方向ステップ数S_xの間隔およびｙ方向ステップ数S_yの間隔で、カーネル２０ごとのデータの畳み込み演算を実施する。ここで、ステップS_x，S_yは１以上の整数である。

【００７４】

図１１は、ＮＮの全結合層であるｌ（ｌ＝１，２，・・・，Ｌ）層目のレイヤにおけるノードごとのエッジの重み情報のマトリクスを示す図である。図１２は、ＮＮの全結合層であるｌ（ｌ＝１，２，・・・，Ｌ）層目のレイヤにおけるノードごとのエッジの重み情報の量子化ステップのマトリクスを示す図である。

【００７５】

ＮＮにおいては、図１１に示すレイヤごとの重みw_{ij}の組み合わせが、ネットワークを構成するデータとなる。このため、ディープニューラルネットワークのような多層のＮＮでは、一般的に数百Ｍｂｙｔｅ以上のデータ量となり、大きなメモリサイズも必要となる。ｉは、ノードインデックスであり、ｉ＝１，２，・・・，N_lである。ｊは、エッジインデックスであり、ｊ＝１，２，・・・，$N_{l-1}+1$（オフセットを含む）である。

【００７６】

そこで、実施の形態１に係るデータ処理装置では、エッジの重み情報のデータ量を削減するため、重み情報を量子化する。例えば、図１２に示すように、量子化ステップq_{ij}は、エッジの重みw_{ij}ごとに設定される。量子化ステップは、複数のノードインデックスまたは複数のエッジインデックスであってもよいし、複数のノードインデックスとエッジインデックスとが共通化されてもよい。これにより、符号化すべき量子化情報が削減され

る。
【００７７】
図１３は、畳み込み層におけるエッジの重み情報のマトリクスを示す図である。図１４は、畳み込み層におけるエッジの重み情報の量子化ステップのマトリクスを示す図である。畳み込み層では、１つのカーネルに対するエッジの重みは、全てのノードで共通であり、ノード一つ当たりに結合するエッジ数、すなわちカーネルサイズＫを小さくしてカーネルを小領域にすることができる。図１３は、エッジの重みｗｉ，ｊがカーネルごとに設定されたデータであり、図１４は、量子化ステップｑｉ，ｊがカーネルごとに設定されたデータである。なお、ｉ’はカーネルインデックスであり、ｉ’＝１，２，・・・，M_l（ｌ＝１，２，・・・，Ｌ）である。ｊ’はエッジインデックスであり、ｊ’＝１，２，・・・，K_l＋１（オフセットを含む）である。
【００７８】
量子化ステップは、複数のカーネルインデックス、複数のエッジインデックス、または複数のカーネルインデックスとエッジインデックスで共通化されてもよい。これにより、符号化すべき量子化情報が削減される。例えば、レイヤ内の全ての量子化ステップを共通化して、一つのレイヤで一つの量子化ステップとしてもよいし、モデル内の全ての量子化ステップを共通化して、一つのモデルで一つの量子化ステップとしてもよい。
【００７９】
図１５は、実施の形態１に係るデータ処理装置（エンコーダ）の変形例の構成を示すブロック図である。図１５に示すデータ処理装置は、学習用データセットと評価用データセットを用いてＮＮを学習し、ＮＮのモデル情報の符号化データを生成する第１のデータ処理装置であり、例えば、図１に示したサーバ１である。図１５に示すデータ処理装置は、データ処理部１０Ａ、符号化部１１および復号部１２を備えている。
【００８０】
データ処理部１０Ａは、ＮＮを生成して学習するデータ処理部であって、学習部１０１Ａ、評価部１０２および制御部１０３を備える。符号化部１１は、学習部１０１によって生成されたモデル情報を符号化し、ヘッダ情報とレイヤ単位の符号化データから構成された符号化データを生成する。復号部１２は、符号化部１１によって生成された符号化データからモデル情報を復号する。また、復号部１２は、復号済みのモデル情報を学習部１０１Ａに出力する。
【００８１】
学習部１０１Ａは、学習部１０１と同様に、学習用データセットを用いてＮＮの学習を実施し、学習されたＮＮの構成を示すモデル情報を生成する。また、学習部１０１Ａは、復号済みのモデル情報を用いてＮＮを生成し、学習用データセットを用いて、生成されたＮＮのパラメータを再学習する。
【００８２】
上記再学習の際、一部のエッジの重みを固定して再学習することで、符号化データのデータサイズを小さく保ったまま高精度化することが可能である。例えば、非零フラグが０の重みは０に固定した状態で再学習を実施することで、再学習前のエッジの重みに係る符号化データのデータサイズ以上となることを防ぎながら重みの最適化が可能となる。
【００８３】
データ処理装置が復号部１２を備え、データ処理部１０Ａが、復号部１２によって復号された情報を用いてＮＮを学習する。これにより、例えば、符号化歪みが発生する非可逆符号化を符号化部１１が行う場合であっても、当該データ処理装置は、符号化データの実際の復号結果に基づいてＮＮを生成して学習することができ、符号化データのデータサイズに対する制約を課した状況下で、符号化誤差の影響を最小限に抑えたＮＮの学習が可能である。
【００８４】
図１と同様の構成を有し、サーバ１として、図３に示したデータ処理装置を備え、クライアント３－１，３－２，・・・，３－Ｎとして、図４に示したデータ処理装置を備えたデ

ータ処理システムにおいて、ＮＮの中間レイヤから出力されるデータは、下記参考文献１に記載された画像検索（ｒｅｔｒｉｅｖａｌ）またはマッチング（ｍａｔｃｈｉｎｇ）を一例とした、画像データおよび音声データに対するデータ処理の特徴量として用いることができる。

（参考文献１）ＩＳＯ／ＩＥＣ ＪＴＣ１／ＳＣ２９／ＷＧ１１／ｍ３９２１９，“Ｉｍｐｒｏｖｅｄ ｒｅｔｒｉｅｖａｌ ａｎｄ ｍａｔｃｈｉｎｇ ｗｉｔｈ ＣＮＮ ｆｅａｔｕｒｅ ｆｏｒ ＣＤＶＡ”，Ｃｈｅｎｇｄｕ，Ｃｈｉｎａ，Ｏｃｔ．２０１６．

【００８５】

例えば、画像検索、マッチングまたは物体追跡といった画像処理の画像特徴量として、ＮＮの中間レイヤの出力データを用いる場合、従来の上記画像処理で用いられていた画像特徴量であるＨＯＧ（Ｈｉｓｔｏｇｒａｍ ｏｆ Ｏｒｉｅｎｔｅｄ Ｇｒａｄｉｅｎｔｓ）、ＳＩＦＴ（Ｓｃａｌｅ Ｉｎｖａｒｉａｎｔ Ｆｅａｔｕｒｅ Ｔｒａｎｓｆｏｒｍ）、または、ＳＵＲＦ（Ｓｐｅｅｄｅｄ Ｕｐ Ｒｏｂｕｓｔ Ｆｅａｔｕｒｅｓ）に対する画像特徴量の置き換えあるいは追加が行われる。これにより、従来の画像特徴量を用いた画像処理と同じ処理手順で当該画像処理を実現できる。実施の形態３に係るデータ処理システムにおいて、符号化部１１は、画像特徴量を出力する中間レイヤまでのＮＮの構成を示すモデル情報を符号化する。

【００８６】

さらに、サーバ１として機能するデータ処理装置は、上記データ処理の特徴量を用いて画像検索等のデータ処理を行う。クライアントとして機能するデータ処理装置は、符号化データから中間レイヤまでのＮＮを生成し、生成されたＮＮの中間レイヤから出力されたデータを特徴量として用いて、画像検索などのデータ処理を実施する。

【００８７】

データ処理システムにおいて、符号化部１１が、ＮＮの中間レイヤまでの構成を示すモデル情報を符号化することによって、量子化によるパラメータデータの圧縮率が高まり、符号化前の重み情報のデータ量を削減することができる。クライアントは、復号部２０１によって復号されたモデル情報を用いてＮＮを生成し、生成されたＮＮの中間レイヤから出力されたデータを特徴量として用いたデータ処理を行う。

【００８８】

また、実施の形態１に係るデータ処理システムは、図１と同様の構成を有し、サーバ１として、図３または図１５に示したデータ処理装置を備え、クライアント３－１，３－２，・・・，３－Ｎとして、図４に示したデータ処理装置を備えることができる。この構成を有したデータ処理システムにおいて、符号化データには、新規レイヤフラグ（ｎｅｗ＿ｌａｙｅｒ＿ｆｌａｇ）が設定されている。新規レイヤフラグが０（無効）である場合は、新規レイヤフラグに対応するレイヤは、参照レイヤを基準として更新されるレイヤである。新規レイヤフラグが１（有効）である場合は、新規レイヤフラグに対応するレイヤは、新規に追加されるレイヤである。

【００８９】

新規レイヤフラグが０（無効）である場合、新規レイヤフラグに対応するレイヤに対してチャネル単位にエッジの重みの更新有無を識別するためのフラグ（ｃｈａｎｎｅｌ＿ｗｉｓｅ＿ｕｐｄａｔｅ＿ｆｌａｇ）が設定される。このフラグが０（無効）であれば、全てのチャネルのエッジの重みが符号化される。このフラグが１（有効）であれば、チャネル単位の重みの更新フラグ（ｃｈａｎｎｅｌ＿ｕｐｄａｔｅ＿ｆｌａｇ）が設定される。この更新フラグは、チャネルごとに参照レイヤからの更新の有無を示すフラグである。この更新フラグが１（有効）である場合、チャネルの重みが符号化され、０（無効）であれば、参照レイヤと同一の重みとされる。

【００９０】

さらに、レイヤ情報ヘッダとして、レイヤのチャネル数を示す情報（ｎｕｍ＿ｃｈａｎｎｅｌｓ）、チャネル単位のエッジの重みの数を示す情報（ｗｅｉｇｈｔｓ＿ｐｅｒ＿ｃｈａｎｎｅｌｓ）が設定される。あるレイヤｌのｗｅｉｇｈｔｓ＿ｐｅｒ＿ｃｈａｎｎｅｌ

ｓは、カーネルサイズＫｌ＋１あるいは一つ前の層であるレイヤｌ－１からのエッジ数Ｎｌ－１＋１となる。

【００９１】

符号化データが、前述した新規レイヤフラグを有することで、レイヤデータユニットの符号化データのみから、チャネル数とチャネル単位の重みの数を特定することが可能である。従って、レイヤデータユニットの復号処理として、チャネル単位の重みの更新フラグを復号することができる。

【００９２】

また、チャネル単位に重みの更新有無を識別するためのフラグが１（有効）に設定される場合は、参照レイヤとチャネル数とが同じであるときに制約される。これは、参照レイヤとチャネル数が異なる場合、参照レイヤと、上記フラグに対応するレイヤとの間で、各チャネルの対応関係が不明になるためである。

【００９３】

図１６は、実施の形態１における符号化データの更新の概要を示す図である。図１６において、上側に示すデータは、非レイヤデータユニットと、レイヤデータユニット（１）～（４）とから構成され、図７と同様に、レイヤデータユニット（４）から順番に符号化される。非レイヤデータユニットには、モデルヘッダ情報として、モデル識別番号（ｍｏｄｅｌ＿ｉｄ）＝０、モデル内レイヤデータユニット数（ｎｕｍ＿ｌａｙｅｒｓ）＝４、レイヤ構造情報（ｍｏｄｅｌ＿ｓｔｒｕｃｔｕｒｅ＿ｉｎｆｏｒｍａｔｉｏｎ）およびレイヤ識別情報（ｌａｙｅｒ＿ｉｄ＿ｉｎｆｏｒｍａｔｉｏｎ）が設定され、符号化データに参照元があるか否かを示すフラグ（ｒｅｆｅｒｅｎｃｅ＿ｍｏｄｅｌ＿ｐｒｅｓｅｎｔ＿ｆｌａｇ）に０（無効）が設定されている。

【００９４】

レイヤデータユニット（１）において、レイヤ識別番号（ｌａｙｅｒ＿ｉｄ）には０が設定され、レイヤのチャネル（フィルタ、カーネル）数を示す情報（ｎｕｍ＿ｃｈａｎｎｅｌｓ）に３２が設定され、チャネル（フィルタ、カーネル）単位の重みの数（バイアス値を含む）を示す情報（ｗｅｉｇｈｔｓ＿ｐｅｒ＿ｃｈａｎｎｅｌｓ）に７６が設定されている。また、レイヤデータユニット（２）において、レイヤ識別番号（ｌａｙｅｒ＿ｉｄ）には１が設定され、レイヤのチャネル数を示す情報（ｎｕｍ＿ｃｈａｎｎｅｌｓ）に６４が設定され、チャネル単位の重みの数を示す情報（ｗｅｉｇｈｔｓ＿ｐｅｒ＿ｃｈａｎｎｅｌｓ）に２８９が設定されている。

【００９５】

レイヤデータユニット（３）において、レイヤ識別番号（ｌａｙｅｒ＿ｉｄ）には２が設定され、レイヤのチャネル数を示す情報（ｎｕｍ＿ｃｈａｎｎｅｌｓ）に１２８が設定され、チャネル単位の重みの数を示す情報（ｗｅｉｇｈｔｓ＿ｐｅｒ＿ｃｈａｎｎｅｌｓ）に５７７が設定されている。また、レイヤデータユニット（４）において、レイヤ識別番号（ｌａｙｅｒ＿ｉｄ）には３が設定され、レイヤのチャネル数を示す情報（ｎｕｍ＿ｃｈａｎｎｅｌｓ）に１００が設定され、チャネル単位の重みの数を示す情報（ｗｅｉｇｈｔｓ＿ｐｅｒ＿ｃｈａｎｎｅｌｓ）に３２７６９が設定されている。

【００９６】

図１６において、下側に示すデータは、レイヤ構造情報、レイヤ更新フラグおよび新規レイヤフラグを用いて、上側に示すデータから更新されたデータであり、非レイヤデータユニットと、レイヤデータユニット（１’）、（２）、（３）、（５）、（４’）とから構成される。上側に示すデータが伝送されたクライアントに対しては、非レイヤデータユニットと、レイヤデータユニット（１’）、（５）、（４’）を送信する必要があるが（Ｎｅｅｄ ｔｏ ｔｒａｎｓｍｉｔ）、レイヤデータユニット（２）および（３）は更新されておらず、送信する必要がない（Ｎｏ ｎｅｅｄ ｔｏ ｔｒａｎｓｍｉｔ）。

【００９７】

図１６の下側に示す非レイヤデータユニットには、モデルヘッダ情報として、モデル識別番号（ｍｏｄｅｌ＿ｉｄ）＝１０、モデル内レイヤデータユニット数（ｎｕｍ＿ｌａｙｅ

ｒｓ）＝５、レイヤ構造情報（ｍｏｄｅｌ＿ｓｔｒｕｃｔｕｒｅ＿ｉｎｆｏｒｍａｔｉｏｎ）およびレイヤ識別情報（ｌａｙｅｒ＿ｉｄ＿ｉｎｆｏｒｍａｔｉｏｎ）が設定され、符号化データに参照元があるか否かを示すフラグ（ｒｅｆｅｒｅｎｃｅ＿ｍｏｄｅｌ＿ｐｒｅｓｅｎｔ＿ｆｌａｇ）に１（有効）が設定され、参照モデル識別番号（ｒｅｆｅｒｅｎｃｅ＿ｍｏｄｅｌ＿ｉｄ）に０が設定され、符号化レイヤデータユニット数（ｎｕｍ＿ｃｏｄｅｄ＿ｌａｙｅｒｓ）に３が設定されている。

【００９８】

レイヤデータユニット（１’）において、レイヤ識別番号（ｌａｙｅｒ＿ｉｄ）は０であり、新規レイヤフラグ（ｎｅｗ＿ｌａｙｅｒ＿ｆｌａｇ）に０が設定され、レイヤのチャネル数を示す情報（ｎｕｍ＿ｃｈａｎｎｅｌｓ）に３２が設定され、チャネル単位の重みの数を示す情報（ｗｅｉｇｈｔｓ＿ｐｅｒ＿ｃｈａｎｎｅｌｓ）に７６が設定されている。また、チャネル単位に重みの更新有無を識別するためのフラグ（ｃｈａｎｎｅｌ＿ｗｉｓｅ＿ｕｐｄａｔｅ＿ｆｌａｇ）には１（有効）が設定されているので、チャネル単位の重みの更新フラグ（ｃｈａｎｎｅｌ＿ｕｐｄａｔｅ＿ｆｌａｇ）が設定されている。

【００９９】

レイヤ識別番号（ｌａｙｅｒ＿ｉｄ）が１であるレイヤデータユニット（２）およびレイヤ識別番号（ｌａｙｅｒ＿ｉｄ）が２であるレイヤデータユニット（３）は、更新対象ではないため、符号化データに含まれない。よって、上記モデルヘッダ情報は、モデル内レイヤデータユニット数（ｎｕｍ＿ｌａｙｅｒｓ）＝５、符号化レイヤデータユニット数（ｎｕｍ＿ｃｏｄｅｄ＿ｌａｙｅｒｓ）＝３が設定されている。

【０１００】

レイヤデータユニット（５）において、レイヤ識別番号（ｌａｙｅｒ＿ｉｄ）は４であり、新規レイヤフラグ（ｎｅｗ＿ｌａｙｅｒ＿ｆｌａｇ）に１（有効）が設定されている。また、レイヤのチャネル数を示す情報（ｎｕｍ＿ｃｈａｎｎｅｌｓ）には２５６が設定され、チャネル単位の重みの数を示す情報（ｗｅｉｇｈｔｓ＿ｐｅｒ＿ｃｈａｎｎｅｌｓ）に１１５３が設定されている。

【０１０１】

レイヤデータユニット（４’）において、レイヤ識別番号（ｌａｙｅｒ＿ｉｄ）は３であり、新規レイヤフラグ（ｎｅｗ＿ｌａｙｅｒ＿ｆｌａｇ）に０が設定され、レイヤのチャネル数を示す情報（ｎｕｍ＿ｃｈａｎｎｅｌｓ）に１００が設定され、チャネル単位の重みの数を示す情報（ｗｅｉｇｈｔｓ＿ｐｅｒ＿ｃｈａｎｎｅｌｓ）に１６３８５が設定されている。また、チャネル単位に重みの更新有無を識別するためのフラグ（ｃｈａｎｎｅｌ＿ｗｉｓｅ＿ｕｐｄａｔｅ＿ｆｌａｇ）には０（無効）が設定され、チャネル単位の重みに更新はない。

【０１０２】

下側に示すデータでは、上側に示すデータにおけるレイヤデータユニット（１）および（４）が、レイヤデータユニット（１’）および（４’）に更新されており、さらに、レイヤ識別番号が４であるレイヤデータユニット（５）が追加されている。

【０１０３】

図１７は、図１６に示す符号化データの更新に対応したネットワークモデルの構成を示す図である。図１７において、左側に示すネットワークモデルが、図１６の上側に示すデータを復号して実現されるネットワークモデルである。また、右側に示すネットワークモデルが、図１６の下側に示すデータを復号して実現されるネットワークモデルである。

【０１０４】

レイヤデータユニット（１’）は、チャネル単位に重みの更新有無を識別するためのフラグ（ｃｈａｎｎｅｌ＿ｗｉｓｅ＿ｕｐｄａｔｅ＿ｆｌａｇ）が１であるので、レイヤデータユニット（１）から、いくつかのチャネルの重みが更新されている。また、レイヤデータユニット（５）が追加され、レイヤデータユニット（４）からレイヤデータユニット（４’）に更新されたことにより、右側に示すネットワークモデルでは、全結合層（Ｆｕｌｌｙ Ｃｏｎｎｅｃｔｅｄ ｌａｙｅｒ）までの間に、２Ｄ畳み込み層（２Ｄ ｃｏｎｖｏｌｕｔ

ｉｏｎ ｌａｙｅｒ）および２Ｄ最大プーリング層（２Ｄ ｍａｘ ｐｏｏｌｉｎｇ ｌａｙｅｒ）が追加されている。

【０１０５】

図１８は、モデル情報ヘッダに含まれるレイヤ構造情報の例を示す図である。モデル情報ヘッダに含まれる全レイヤ構造情報（ｍｏｄｅｌ＿ｓｔｒｕｃｔｕｒｅ＿ｉｎｆｏｒｍａｔｉｏｎ）として、図１８に示すようなテキスト情報を設定してもよい。図１８に示すテキスト情報は、ＮＮＥＦ（Ｎｅｕｒａｌ Ｎｅｔｗｏｒｋ Ｅｘｃｈａｎｇｅ Ｆｏｒｍａｔ）という、参考文献２に記載される標準規格によるモデルのレイヤ構造を示すテキスト情報である。

（参考文献２）“Ｎｅｕｒａｌ Ｎｅｔｗｏｒｋ Ｅｘｃｈａｎｇｅ Ｆｏｒｍａｔ”, Ｔｈｅ Ｋｈｒｏｎｏｓ ＮＮＥＦ Ｗｏｒｋｉｎｇ Ｇｒｏｕｐ, Ｖｅｒｓｉｏｎ １．０, Ｒｅｖｉｓｉｏｎ ３, ２０１８－０６－１３．

【０１０６】

図１８において、（Ａ）ｍｏｄｅｌ＿ｉｄ＝０のネットワークモデルは、図１６の上側に示したデータに対応するネットワークモデル（図１７の左側に示したネットワークモデル）である。（Ｂ）ｍｏｄｅｌ＿ｉｄ＝１０のネットワークモデルは、図１６の下側に示したデータに対応するネットワークモデル（図１７の右側に示したネットワークモデル）である。

【０１０７】

図１９は、モデル情報ヘッダに含まれるレイヤ構造情報に対応するレイヤ識別情報（ｌａｙｅｒ＿ｉｄ＿ｉｎｆｏｒｍａｔｉｏｎ）の例を示す図であり、図１８のレイヤ構造情報に対応するレイヤ識別番号が設定されたレイヤ識別情報を示している。図１９において、（Ａ）ｍｏｄｅｌ＿ｉｄ＝０のネットワークモデルは、図１７の左側に示したネットワークモデルに対応するレイヤ識別情報である。（Ｂ）ｍｏｄｅｌ＿ｉｄ＝１０のネットワークモデルは、図１７の右側に示したネットワークモデルに対応するレイヤ識別情報である。各レイヤの重みおよびバイアス値がレイヤ識別番号に割り当てられ、その値は図１６に示したデータに対応している。

【０１０８】

全レイヤ構造情報であるｍｏｄｅｌ＿ｓｔｒｕｃｔｕｒｅ＿ｉｎｆｏｒｍａｔｉｏｎと全レイヤ構造情報に対応するレイヤ識別番号を示す情報であるｌａｙｅｒ＿ｉｄ＿ｉｎｆｏｒｍａｔｉｏｎが記載されたファイル等のファイルデータは、モデル情報ヘッダにおいて、それぞれ上記ファイルデータのバイト数を示す情報の後に挿入する形で符号化データを構成する。あるいは、上記ファイルデータの入手先を示すＵＲＬ（Ｕｎｉｆｏｒｍ Ｒｅｓｏｕｒｃｅ Ｌｏｃａｔｏｒ）をモデル情報ヘッダに含める構成も可能である。さらに、これらの構成のいずれかを選択できるように、いずれの構成であるかを識別するフラグを、モデル情報ヘッダにおける上記ファイルデータまたはＵＲＬの前に設定してもよい。上記識別フラグは、ｍｏｄｅｌ＿ｓｔｒｕｃｔｕｒｅ＿ｉｎｆｏｒｍａｔｉｏｎとｌａｙｅｒ＿ｉｄ＿ｉｎｆｏｒｍａｔｉｏｎとで共通でもよいし、個別に持つようにしてもよい。前者であれば、モデル情報ヘッダの情報量を削減でき、後者であれば、使用する際の前提条件に応じて独立に設定できる。

さらに、モデル情報ヘッダは、上記テキスト情報のフォーマットを示す情報を含む。例えば、ＮＮＥＦはインデックス０、その他のフォーマットが１以降となるような情報である。これによって、どのフォーマットで記述されているかを識別することができ、正しく復号することができる。

【０１０９】

なお、図１８および図１９に示したようなテキスト情報で表されたレイヤ構造情報と、レイヤ構造情報に対応するレイヤ識別番号を示す情報は、実施の形態１で示した全てのシステムに適用することが可能である。さらに、ｍｏｄｅｌ＿ｓｔｒｕｃｔｕｒｅ＿ｉｎｆｏｒｍａｔｉｏｎとｌａｙｅｒ＿ｉｄ＿ｉｎｆｏｒｍａｔｉｏｎから、当該符号化データのみから各レイヤデータユニットがモデル内のどのレイヤのデータであるのかを識別するこ

とができる。したがって、モデルを更新する場合（ｒｅｆｅｒｅｎｃｅ＿ｍｏｄｅｌ＿ｐｒｅｓｅｎｔ＿ｆｌａｇが有効である場合）、本実施の形態で示す符号化データから生成されていないモデルを参照モデルとすることも可能である。すなわち、本実施の形態で示す符号化データは、モデル情報ヘッダの一部としてｍｏｄｅｌ＿ｓｔｒｕｃｔｕｒｅ＿ｉｎｆｏｒｍａｔｉｏｎとｌａｙｅｒ＿ｉｄ＿ｉｎｆｏｒｍａｔｉｏｎを持つことで、任意のモデルを参照モデルとして設定することができる。ただし、この場合、参照モデル識別番号（ｒｅｆｅｒｅｎｃｅ＿ｍｏｄｅｌ＿ｉｄ）と参照モデルとの対応付けは別途定義しておく必要がある。

【０１１０】

次に、実施の形態１に係るデータ処理装置の機能を実現するハードウェア構成について説明する。実施の形態１に係るデータ処理装置における、データ処理部１０および符号化部１１の機能は、処理回路により実現される。すなわち、実施の形態１に係るデータ処理装置は、図５のステップＳＴ１からステップＳＴ６までの処理を実行するための処理回路を備える。処理回路は、専用のハードウェアであってもよいが、メモリに記憶されたプログラムを実行するＣＰＵ（Ｃｅｎｔｒａｌ Ｐｒｏｃｅｓｓｉｎｇ Ｕｎｉｔ）であってもよい。

【０１１１】

図２０Ａは、実施の形態１に係るデータ処理装置の機能を実現するハードウェア構成を示すブロック図である。図２０Ａにおいて、処理回路３００は、図３に示したデータ処理装置として機能する専用の回路である。図２０Ｂは、実施の形態１に係るデータ処理装置の機能を実現するソフトウェアを実行するハードウェア構成を示すブロック図である。図２０Ｂにおいて、プロセッサ３０１およびメモリ３０２は、信号バスによって互いに接続されている。

【０１１２】

上記処理回路が図２０Ａに示す専用のハードウェアである場合、処理回路３００は、例えば、単一回路、複合回路、プログラム化したプロセッサ、並列プログラム化したプロセッサ、ＡＳＩＣ（Ａｐｐｌｉｃａｔｉｏｎ Ｓｐｅｃｉｆｉｃ Ｉｎｔｅｇｒａｔｅｄ Ｃｉｒｃｕｉｔ）、ＦＰＧＡ（Ｆｉｅｌｄ－Ｐｒｏｇｒａｍｍａｂｌｅ Ｇａｔｅ Ａｒｒａｙ）またはこれらを組み合わせたものが該当する。なお、データ処理部１０および符号化部１１の機能を別々の処理回路で実現してもよいし、これらの機能をまとめて１つの処理回路で実現してもよい。

【０１１３】

上記処理回路が図２０Ｂに示すプロセッサである場合、データ処理部１０および符号化部１１の機能は、ソフトウェア、ファームウェアまたはソフトウェアとファームウェアとの組み合わせによって実現される。ソフトウェアまたはファームウェアは、プログラムとして記述されて、メモリ３０２に記憶される。プロセッサ３０１は、メモリ３０２に記憶されたプログラムを読み出して実行することによって、データ処理部１０および符号化部１１の機能を実現する。すなわち、実施の形態１に係るデータ処理装置は、プロセッサ３０１によって実行されるときに、図５に示したステップＳＴ１からステップＳＴ６までの処理が結果的に実行されるプログラムを記憶するためのメモリ３０２を備える。これらのプログラムは、データ処理部１０および符号化部１１の手順または方法をコンピュータに実行させるものである。メモリ３０２は、コンピュータを、データ処理部１０および符号化部１１として機能させるためのプログラムが記憶されたコンピュータ可読記憶媒体であってもよい。

【０１１４】

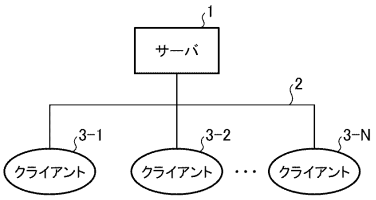
メモリ３０２には、例えば、ＲＡＭ（Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）、ＲＯＭ（Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）、フラッシュメモリ、ＥＰＲＯＭ（Ｅｒａｓａｂｌｅ Ｐｒｏｇｒａｍｍａｂｌｅ Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）、ＥＥＰＲＯＭ（Ｅｌｅｃｔｒｉｃａｌｌｙ－ＥＰＲＯＭ）などの不揮発性または揮発性の半導体メモリ、磁気ディスク、フレキシブルディスク、光ディスク、コンパクトディスク、ミニディスク

、ＤＶＤなどが該当する。
【０１１５】
なお、データ処理部１０および符号化部１１の機能について一部を専用のハードウェアで実現し、一部をソフトウェアまたはファームウェアで実現してもよい。例えば、データ処理部１０については、専用のハードウェアとしての処理回路でその機能を実現し、符号化部１１については、プロセッサ３０１がメモリ３０２に記憶されたプログラムを読み出して実行することによってその機能を実現してもよい。このように、処理回路は、ハードウェア、ソフトウェア、ファームウェアまたはこれらの組み合わせによって、上記機能のそれぞれを実現することができる。
【０１１６】
なお、図３に示したデータ処理装置について説明したが、図４に示したデータ処理装置においても、同様である。例えば、図４に示したデータ処理装置は、図６のステップＳＴ１１からステップＳＴ１３までの処理を実行するための処理回路を備える。この処理回路は、専用のハードウェアであってもよいが、メモリに記憶されたプログラムを実行するＣＰＵであってもよい。
【０１１７】
上記処理回路が図２０Ａに示す専用のハードウェアである場合、処理回路３００は、例えば、単一回路、複合回路、プログラム化したプロセッサ、並列プログラム化したプロセッサ、ＡＳＩＣ、ＦＰＧＡまたはこれらを組み合わせたものが該当する。なお、復号部２０１および推論部２０２の機能を、別々の処理回路で実現してもよいし、これらの機能をまとめて１つの処理回路で実現してもよい。
【０１１８】
上記処理回路が図２０Ｂに示すプロセッサであると、復号部２０１および推論部２０２の機能は、ソフトウェア、ファームウェアまたはソフトウェアとファームウェアとの組み合わせによって実現される。ソフトウェアまたはファームウェアは、プログラムとして記述されて、メモリ３０２に記憶される。プロセッサ３０１は、メモリ３０２に記憶されたプログラムを読み出して実行することによって、復号部２０１および推論部２０２の機能を実現する。すなわち、図４に示したデータ処理装置は、プロセッサ３０１によって実行されるときに、図６に示すステップＳＴ１１からステップＳＴ１３までの処理が結果的に実行されるプログラムを記憶するためのメモリ３０２を備える。これらのプログラムは、復号部２０１および推論部２０２の手順または方法を、コンピュータに実行させるものである。メモリ３０２は、コンピュータを、復号部２０１および推論部２０２として機能させるためのプログラムが記憶されたコンピュータ可読記憶媒体であってもよい。
【０１１９】
なお、復号部２０１および推論部２０２の機能について一部を専用のハードウェアで実現し、一部をソフトウェアまたはファームウェアで実現してもよい。例えば、復号部２０１については専用のハードウェアとしての処理回路でその機能を実現し、推論部２０２については、プロセッサ３０１がメモリ３０２に記憶されたプログラムを読み出して実行することによってその機能を実現してもよい。
【０１２０】
以上のように、実施の形態１に係るデータ処理装置において、符号化部１１が、レイヤ構造情報を符号化し、レイヤ更新フラグを符号化し、レイヤ更新フラグがレイヤ構造の更新を示す場合は、新規レイヤフラグを符号化する。ＮＮの構造を示すデータのうち、更新されたレイヤに関する情報のみが符号化されて伝送されるので、ＮＮの構造を示すデータのデータサイズを削減することができる。
【０１２１】
また、符号化部１１が、ＮＮの構成を示す情報を符号化し、ヘッダ情報とレイヤ単位の符号化データから構成された符号化データを生成する。復号側で必要なレイヤに関する情報のみを符号化することができるので、ＮＮの構成に関する情報を符号化する処理負荷が低減され、復号側へ伝送するデータサイズを削減することができる。

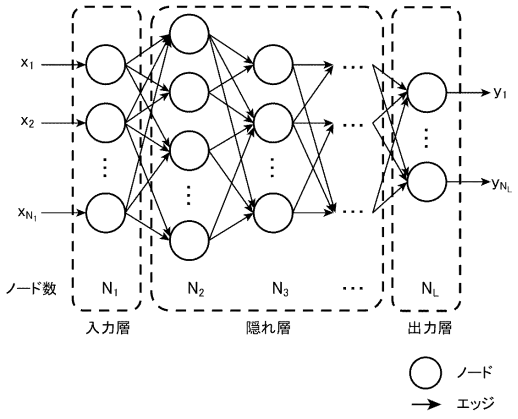
【０１２２】
実施の形態１に係るデータ処理装置において、符号化部１１が、ＮＮのレイヤに属するエッジの重み情報を、上位ビットからビットプレーン単位で符号化する。これにより、復号側へ伝送する符号化データのデータサイズを削減することができる。
【０１２３】
実施の形態１に係るデータ処理装置において、符号化部１１が、ヘッダ情報で指定された１以上のレイヤに関する情報を符号化する。これにより、復号側で必要なレイヤに関する情報のみが符号化され、復号側へ伝送する符号化データのデータサイズを削減することができる。
【０１２４】
実施の形態１に係るデータ処理装置において、符号化部１１が、ヘッダ情報で指定されたレイヤに属するエッジの重みの値と特定の値との差分を符号化する。これにより、復号側へ伝送する符号化データのデータサイズを削減することができる。
【０１２５】
実施の形態１に係るデータ処理装置において、符号化部１１が、エッジの重み情報を、ベース符号化データとエンハンスメント符号化データに分けて符号化する。これにより、データ伝送ネットワーク２の伝送帯域と伝送許容時間に応じた符号化データの伝送を実現することができる。
【０１２６】
なお、本発明は上記実施の形態に限定されるものではなく、本発明の範囲内において、実施の形態のそれぞれの自由な組み合わせまたは実施の形態のそれぞれの任意の構成要素の変形もしくは実施の形態のそれぞれにおいて任意の構成要素の省略が可能である。
【産業上の利用可能性】
【０１２７】
本発明に係るデータ処理装置は、例えば、画像認識技術への利用が可能である。
【符号の説明】
【０１２８】
１　サーバ、２　データ伝送ネットワーク、３－１～３－Ｎ　クライアント、１０，１０Ａ　データ処理部、１０－１～１０－９，１１－１～１１－３　ノード、１１　符号化部、１２　復号部、１２－１～１２－１５　エッジ、２０　カーネル、１０１，１０１Ａ　学習部、１０２　評価部、１０３　制御部、２０１　復号部、２０２　推論部、３００　処理回路、３０１　プロセッサ、３０２　メモリ。

【図面】

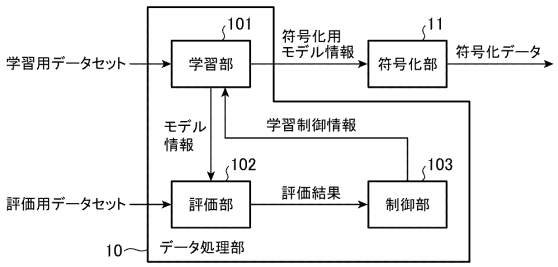
【図１】


1
サーバ
2
3-1
3-2
3-N
クライアント
クライアント
クライアント

【図２】


x1
x2
xN1
y1
yNL
ノード数
N1
N2
N3
NL
入力層
隠れ層
出力層
ノード
エッジ

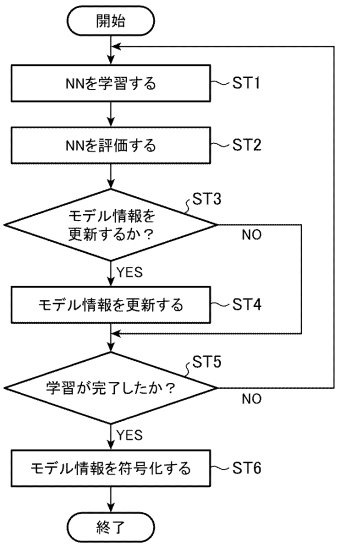
【図３】


101
学習部
符号化用
モデル情報
11
符号化部
符号化データ
学習用データセット
モデル
情報
学習制御情報
102
103
評価用データセット
評価部
評価結果
制御部
10
データ処理部

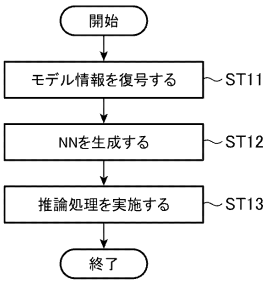
【図４】


評価用データ
201
202
符号化データ
復号部
モデル情報
推論部
推論結果

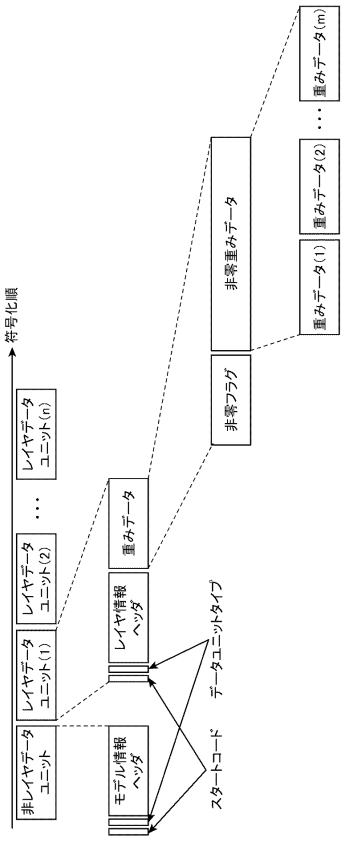
【図５】


開始
NNを学習する
ST1
NNを評価する
ST2
モデル情報を
更新するか？
ST3
NO
YES
モデル情報を更新する
ST4
学習が完了したか？
ST5
NO
YES
モデル情報を符号化する
ST6
終了

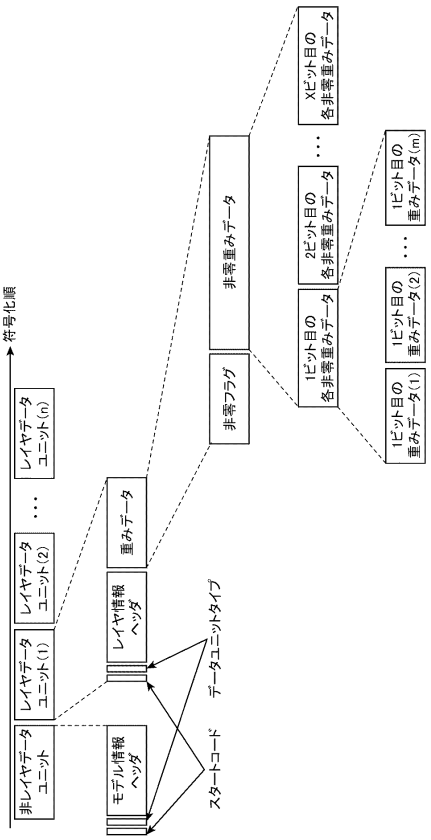
【図６】


開始
モデル情報を復号する
ST11
NNを生成する
ST12
推論処理を実施する
ST13
終了

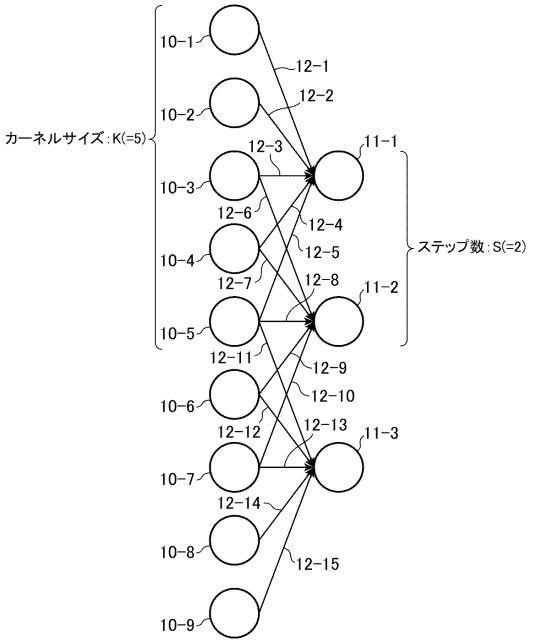
【図７】


符号化順
非レイヤデータユニット
レイヤデータユニット(1)
レイヤデータユニット(2)
…
レイヤデータユニット(n)
モデル情報ヘッダ
レイヤ情報ヘッダ
重みデータ
非零フラグ
非零重みデータ
重みデータ(1)
重みデータ(2)
…
重みデータ(m)
スタートコード
データユニットタイプ

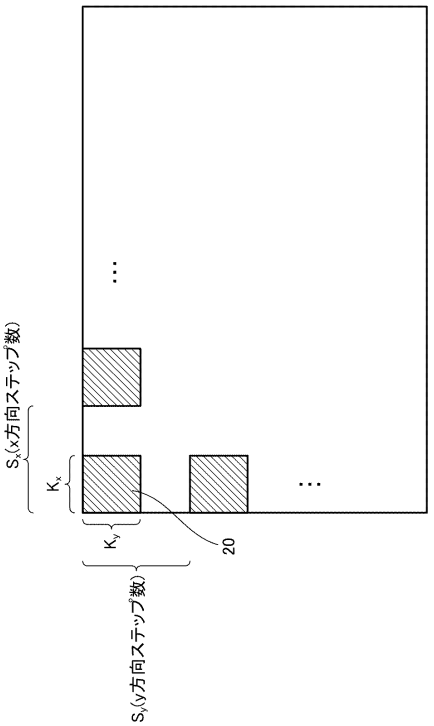
【図８】


符号化順
非レイヤデータユニット
レイヤデータユニット(1)
レイヤデータユニット(2)
…
レイヤデータユニット(n)
モデル情報ヘッダ
レイヤ情報ヘッダ
重みデータ
非零フラグ
非零重みデータ
1ビット目の各非零重みデータ
2ビット目の各非零重みデータ
…
Xビット目の各非零重みデータ
1ビット目の重みデータ(1)
1ビット目の重みデータ(2)
…
1ビット目の重みデータ(m)
スタートコード
データユニットタイプ

【図９】



【図１０】



【図１１】

		ノードインデックス				
		1	2	3	…	N_l
エッジインデックス	1	w_{11}	w_{21}	w_{31}	…	w_{N_l1}
	2	w_{12}	w_{22}	w_{32}	…	w_{N_l2}
	3	w_{13}	w_{23}	w_{33}	…	w_{N_l3}
	⋮	⋮	⋮	⋮	…	⋮
	N_{l-1}	$w_{1N_{l-1}}$	$w_{2N_{l-1}}$	$w_{3N_{l-1}}$	…	$w_{N_lN_{l-1}}$
	$N_{l-1}+1$	$w_{1N_{l-1}+1}$	$w_{2N_{l-1}+1}$	$w_{3N_{l-1}+1}$	…	$w_{N_lN_{l-1}+1}$

【図１２】

		ノードインデックス				
		1	2	3	…	N_l
エッジインデックス	1	q_{11}	q_{21}	q_{31}	…	q_{N_l1}
	2	q_{12}	q_{22}	q_{32}	…	q_{N_l2}
	3	q_{13}	q_{23}	q_{33}	…	q_{N_l3}
	⋮	⋮	⋮	⋮	…	⋮
	N_{l-1}	$q_{1N_{l-1}}$	$q_{2N_{l-1}}$	$q_{3N_{l-1}}$	…	$q_{N_lN_{l-1}}$
	$N_{l-1}+1$	$q_{1N_{l-1}+1}$	$q_{2N_{l-1}+1}$	$q_{3N_{l-1}+1}$	…	$q_{N_lN_{l-1}+1}$

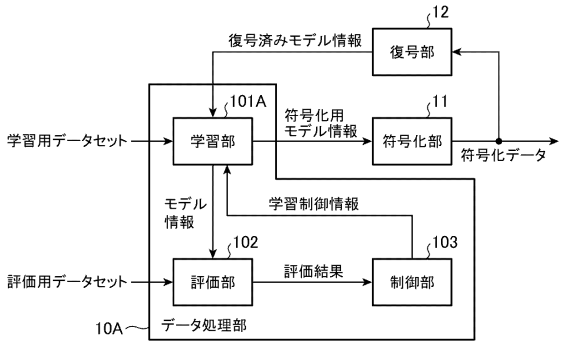
【図１３】

		カーネルインデックス				
		1	2	3	…	M_l
エッジインデックス	1	w_{11}	w_{21}	w_{31}	…	w_{M_l1}
	2	w_{12}	w_{22}	w_{32}	…	w_{M_l2}
	3	w_{13}	w_{23}	w_{33}	…	w_{M_l3}
	⋮	⋮	⋮	⋮	…	⋮
	K_l	w_{1K_l}	w_{2K_l}	w_{3K_l}	…	$w_{M_lK_l}$
	K_l+1	w_{1K_l+1}	w_{2K_l+1}	w_{3K_l+1}	…	$w_{M_lK_l+1}$

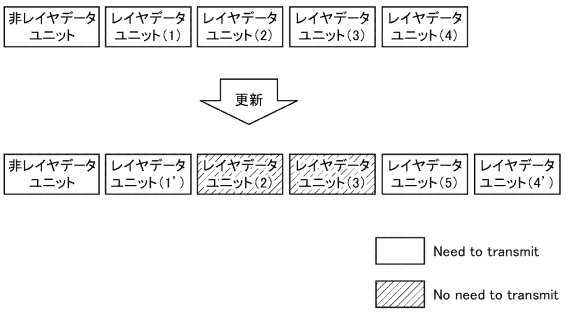
【図１４】

		カーネルインデックス				
		1	2	3	…	M_l
エッジインデックス	1	q_{11}	q_{21}	q_{31}	…	q_{M_l1}
	2	q_{12}	q_{22}	q_{32}	…	q_{M_l2}
	3	q_{13}	q_{23}	q_{33}	…	q_{M_l3}
	⋮	⋮	⋮	⋮	…	⋮
	K_l	q_{1K_l}	q_{2K_l}	q_{3K_l}	…	$q_{M_lK_l}$
	K_l+1	q_{1K_l+1}	q_{2K_l+1}	q_{3K_l+1}	…	$q_{M_lK_l+1}$

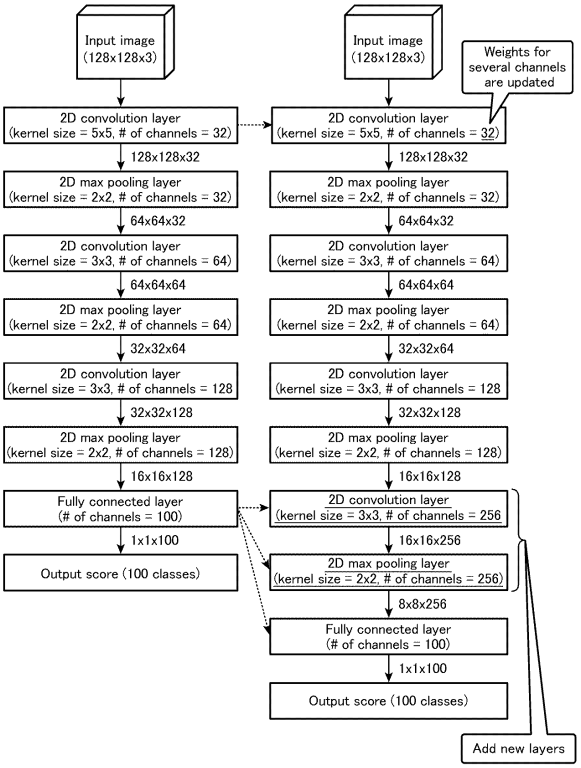
【図１５】



【図１６】



【図１７】



【図１８】

(A) model_id = 0のネットワークモデル

```
graph ExampleNet( input ) -> ( output )
{
    input = external(shape = [1, 3, 128, 128])
    kernel1 = variable(shape = [32, 3, 5, 5], label = 'examplenet/conv1/kernel')
    bias1 = variable(shape = [1, 32], label = 'examplenet/conv1/bias')
    conv1 = conv(input, kernel1, bias1, padding = [(1,1), (1,1)],
                 border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu1 = relu(conv1)
    pool1 = max_pool(relu1, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
                     border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel2 = variable(shape = [64, 32, 3, 3], label = 'examplenet/conv2/kernel')
    bias2 = variable(shape = [1, 64], label = 'examplenet/conv2/bias')
    conv2 = conv(pool1, kernel2, bias2, padding = [(1,1), (1,1)],
                 border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu2 = relu(conv2)
    pool2 = max_pool(relu2, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
                     border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel3 = variable(shape = [128, 64, 3, 3], label = 'examplenet/conv3/kernel')
    bias3 = variable(shape = [1, 128], label = 'examplenet/conv3/bias')
    conv3 = conv(pool2, kernel3, bias3, padding = [(1,1), (1,1)],
                 border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu3 = relu(conv3)
    pool3 = max_pool(relu3, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
                     border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel4 = variable(shape = [256, 128, 3, 3], label = 'examplenet/conv4/kernel')
    bias4 = variable(shape = [1, 256], label = 'examplenet/conv4/bias')
    conv4 = conv(pool3, kernel4, bias4, padding = [(1,1), (1,1)],
                 border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu4 = relu(conv4)
    pool4 = max_pool(relu4, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
                     border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel5 = variable(shape = [100, 256, 8, 8], label = 'examplenet/fc5/kernel')
    bias5 = variable(shape = [1, 100], label = 'examplenet/fc5/bias')
    conv5 = conv(pool4, kernel5, bias5, padding = [(0,0), (0,0)],
                 border = 'constant', stride = [1, 1], dilation = [1, 1])
    output = softmax(conv5)
}
```

(B) model_id = 10のネットワークモデル

```
graph ExampleNet( input ) -> ( output )
{
    input = external(shape = [1, 3, 128, 128])
    kernel1 = variable(shape = [32, 3, 5, 5], label = 'examplenet/conv1/kernel')
    bias1 = variable(shape = [1, 32], label = 'examplenet/conv1/bias')
    conv1 = conv(input, kernel1, bias1, padding = [(1,1), (1,1)],
                 border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu1 = relu(conv1)
    pool1 = max_pool(relu1, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
                     border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel2 = variable(shape = [64, 32, 3, 3], label = 'examplenet/conv2/kernel')
    bias2 = variable(shape = [1, 64], label = 'examplenet/conv2/bias')
    conv2 = conv(pool1, kernel2, bias2, padding = [(1,1), (1,1)],
                 border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu2 = relu(conv2)
    pool2 = max_pool(relu2, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
                     border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel3 = variable(shape = [128, 64, 3, 3], label = 'examplenet/conv3/kernel')
    bias3 = variable(shape = [1, 128], label = 'examplenet/conv3/bias')
    conv3 = conv(pool2, kernel3, bias3, padding = [(1,1), (1,1)],
                 border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu3 = relu(conv3)
    pool3 = max_pool(relu3, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
                     border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel4 = variable(shape = [256, 128, 3, 3], label = 'examplenet/conv4/kernel')
    bias4 = variable(shape = [1, 256], label = 'examplenet/conv4/bias')
    conv4 = conv(pool3, kernel4, bias4, padding = [(1,1), (1,1)],
                 border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu4 = relu(conv4)
    pool4 = max_pool(relu4, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
                     border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel5 = variable(shape = [100, 256, 8, 8], label = 'examplenet/fc5/kernel')
    bias5 = variable(shape = [1, 100], label = 'examplenet/fc5/bias')
    conv5 = conv(pool4, kernel5, bias5, padding = [(0,0), (0,0)],
                 border = 'constant', stride = [1, 1], dilation = [1, 1])
    output = softmax(conv5)
}
```

【図１９】

(A) model_id = 0のネットワークモデル

```
layer ExampleNet( input ) -> ( output )
{
    0 = kernel1, bias1
    1 = kernel2, bias2
    2 = kernel3, bias3
    3 = kernel4, bias4
}
```

(B) model_id = 10のネットワークモデル

```
layer ExampleNet( input ) -> ( output )
{
    0 = kernel1, bias1
    1 = kernel2, bias2
    2 = kernel3, bias3
    3 = kernel5, bias5
    4 = kernel4, bias4
}
```

【図２０】

図20A

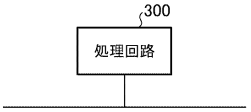
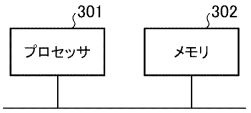


図20B



フロントページの続き

(56)参考文献　　国際公開第２０１９／００８７５２（WO，Ａ１）
　　国際公開第２０１６／１９９３３０（WO，Ａ１）
(58)調査した分野　(Int.Cl.，ＤＢ名)
　　Ｇ０６Ｎ　３／００－２０／００