(19)日本国特許庁(JP)

# (12)特許公報(B2)

(11)特許番号
特許第7549153号
(P7549153)

(45)発行日　令和6年9月10日(2024.9.10)

(24)登録日　令和6年9月2日(2024.9.2)

| (51)国際特許分類 | | FI | |
|---|---|---|---|
| *G06N* | *3/10 (2006.01)* | G06N | 3/10 |
| *G06F* | *8/40 (2018.01)* | G06F | 8/40 |
| *G06F* | *8/30 (2018.01)* | G06F | 8/30 |

請求項の数　35　（全31頁）

(21)出願番号　特願2023-549884(P2023-549884)
(86)(22)出願日　令和4年11月2日(2022.11.2)
(65)公表番号　特表2024-527451(P2024-527451A)
(43)公表日　令和6年7月25日(2024.7.25)
(86)国際出願番号　PCT/CN2022/129228
(87)国際公開番号　WO2023/221408
(87)国際公開日　令和5年11月23日(2023.11.23)
審査請求日　令和5年8月23日(2023.8.23)
(31)優先権主張番号　202210560831.9
(32)優先日　令和4年5月19日(2022.5.19)
(33)優先権主張国・地域又は機関
中国(CN)

(73)特許権者　514322098
ベイジン バイドゥ ネットコム サイエンス テクノロジー カンパニー リミテッド
Ｂｅｉｊｉｎｇ Ｂａｉｄｕ Ｎｅｔｃｏｍ Ｓｃｉｅｎｃｅ Ｔｅｃｈｎｏｌｏｇｙ Ｃｏ．，Ｌｔｄ．
中華人民共和国 ペキン １０００８５，ハイディアン ディストリクト，シャンディ テンス ストリート，１０番，バイドゥ キャンパス ２階
２／Ｆ Ｂａｉｄｕ Ｃａｍｐｕｓ，Ｎｏ．１０，Ｓｈａｎｇｄｉ １０ｔｈ Ｓｔｒｅｅｔ，Ｈａｉｄｉａｎ Ｄｉｓｔｒｉｃｔ，Ｂｅｉｊｉｎｇ １０００

最終頁に続く

(54)【発明の名称】　深層学習フレームワークの演算子処理方法及び装置、電子機器、記憶媒体ならびにコンピュータプログラム

(57)【特許請求の範囲】
【請求項１】
深層学習フレームワークの演算子処理方法であって、
演算子カーネル関数、及び深層学習フレームワークに依存しないテンプレートパラメータを含む処理対象演算子を取得することと、
前記処理対象演算子に対する入力情報を受信したことに応答して、前記入力情報を用いて前記テンプレートパラメータを解析して、深層学習フレームワークに依存する複数の完全なテンプレートパラメータを得ることと、
複数の完全なテンプレートパラメータに基づいて、前記演算子カーネル関数を処理して、前記深層学習フレームワーク用の利用可能な演算子を得ることと、を含む
深層学習フレームワークの演算子処理方法。
【請求項２】
前記処理対象演算子に対する入力情報を受信したことに応答して、前記入力情報を用いて前記テンプレートパラメータを解析し、深層学習フレームワークに依存する複数の完全なテンプレートパラメータを得ることは、
マクロを用いて可変パラメータリストを作成することと、
前記可変パラメータリストに基づいて前記テンプレートパラメータを反復に解析することと、を含み、
ここで、前記入力情報を前記マクロのマクロパラメータとする
請求項１に記載の方法。

【請求項３】
前記可変パラメータリストは、
処理対象演算子の名称と、
前記演算子カーネル関数の名称と、
前記演算子カーネル関数のデータレイアウトと、
前記深層学習フレームワークを実装するための機器タイプと、
前記深層学習フレームワークを実装するためのデータタイプと、を含む
請求項２に記載の方法。
【請求項４】
前記可変パラメータリストに基づいて前記テンプレートパラメータを反復に解析することは、
前記マクロに含まれるパラメータ数を特定することと、
前記パラメータ数に応じて、反復終了位置を特定することと、
可変パラメータリストを用いて、前記反復終了位置に応じて前記入力情報を反復に処理することと、を含む
請求項２に記載の方法。
【請求項５】
前記可変パラメータリストに基づいて前記テンプレートパラメータを反復に解析することは、
前記可変パラメータリストを用いて前記テンプレートパラメータを反復に解析して、複数の内部テンプレートパラメータを得ることと、
前記複数の内部テンプレートパラメータをスプライシングして、前記複数の完全なテンプレートパラメータを得ることと、をさらに含む
請求項２に記載の方法。
【請求項６】
前記複数の内部テンプレートパラメータは複数の文字列を含み、前記複数の内部テンプレートパラメータをスプライシングして、前記複数の完全なテンプレートパラメータを得ることは、
前記入力情報に基づいて、前記複数の文字列をスプライシングして、前記処理対象演算子に対する複数の完全なテンプレートパラメータを得ることを含む
請求項５に記載の方法。
【請求項７】
複数の完全なテンプレートパラメータに基づいて、前記演算子カーネル関数を処理して、前記深層学習フレームワーク用の利用可能な演算子を得ることは、
前記複数の完全なテンプレートパラメータを明示的にインスタンス化することで、前記演算子カーネル関数を処理して、前記深層学習フレームワーク用の利用可能な演算子を得ることを含む
請求項５に記載の方法。
【請求項８】
前記複数の完全なテンプレートパラメータの各々は、テンプレート関数記述情報及びテンプレート関数情報を含み、前記複数の完全なテンプレートパラメータを明示的にインスタンス化することで、前記演算子カーネル関数を処理して、前記深層学習フレームワーク用の利用可能な演算子を得ることは、
前記テンプレート関数記述情報をヘッダファイルとしてインスタンス化することと、
前記テンプレート関数情報をソースコードファイルとしてインスタンス化することと、を含む
請求項７に記載の方法。
【請求項９】
前記複数の完全なテンプレートパラメータに基づいて、前記演算子カーネル関数を処理して、前記深層学習フレームワーク用の利用可能な演算子を得ることは、

前記複数の完全なテンプレートパラメータのうちの各完全なテンプレートパラメータに対して、該完全なテンプレートパラメータの関数パラメータタイプを特定することと、
前記関数パラメータタイプに基づいて、前記完全なテンプレートパラメータの入力情報を対応するパラメータ情報に変換することと、
前記対応するパラメータ情報を記録することと、を含む
請求項５に記載の方法。
【請求項１０】
前記複数の完全なテンプレートパラメータに基づいて、前記深層学習フレームワークに基づく登録対象演算子を作成することと、
前記登録対象演算子を前記深層学習フレームワーク内部のグローバル演算子テーブルに登録することと、をさらに含む
請求項５に記載の方法。
【請求項１１】
前記登録対象演算子は登録対象演算子記述情報及び登録対象演算子カーネル関数を含み、
前記登録対象演算子カーネル関数には正規化された入力情報及び正規化された関数ポインタが含まれ、前記複数の完全な内部テンプレート関数に基づいて、前記深層学習フレームワークに基づく登録対象演算子を作成することは、
前記複数の完全なテンプレートパラメータのうちの各完全なテンプレートパラメータに対して、前記完全なテンプレートパラメータに対応する、正規化された形式を有する静的関数を含む構造体を特定することと、
前記静的関数の入力情報を前記正規化された入力情報とすることと、
前記静的関数の関数ポインタを前記正規化された関数ポインタとすることと、を含む
請求項１０に記載の方法。
【請求項１２】
前記静的関数の入力情報は入力情報リストを構成し、前記入力情報リストは入力テンソルリストを含み、前記構造体は少なくとも１つの特化サブ構造体を含み、前記少なくとも１つの特化サブ構造体の各々は対応するデータタイプを有する
請求項１１に記載の方法。
【請求項１３】
前記グローバル演算子テーブルは、
演算子カーネル工場クラスと、
演算子カーネル名称クラス及びカーネルキー値クラスと、
演算子カーネルクラスと、を含む
請求項１０に記載の方法。
【請求項１４】
前記演算子カーネルクラスは、
演算子カーネル関数のポインタと、
演算子カーネル関数入力パラメータの記述情報と、
演算子カーネル関数出力パラメータの記述情報と、の少なくとも１つを含む
請求項１３に記載の方法。
【請求項１５】
前記深層学習フレームワークに依存しないテンプレートパラメータは、前記処理対象演算子を配布するために用いられ、前記深層学習フレームワークに依存しないテンプレートパラメータは、前記深層学習フレームワークの実装機器及び前記深層学習フレームワークのデータタイプのいずれにも依存しないテンプレートパラメータを含む
請求項１に記載の方法。
【請求項１６】
前記利用可能な演算子に対する呼び出しコマンドに応答して、前記利用可能な演算子を呼び出すことと、
呼び出された前記利用可能な演算子をコンパイルすることで、前記利用可能な演算子に対

応する関数機能を実行することと、をさらに含む
請求項１に記載の方法。
【請求項１７】
深層学習フレームワークの演算子処理装置であって、
演算子カーネル関数、及び深層学習フレームワークに依存しないテンプレートパラメータを含む処理対象演算子を取得するための取得モジュールと、
前記処理対象演算子に対する入力情報を受信したことに応答して、前記入力情報を用いて前記テンプレートパラメータを解析して、深層学習フレームワークに依存する複数の完全なテンプレートパラメータを得るための解析モジュールと、
複数の完全なテンプレートパラメータに基づいて、前記演算子カーネル関数を処理して、前記深層学習フレームワーク用の利用可能な演算子を得るための処理モジュールと、を含む
深層学習フレームワークの演算子処理装置。
【請求項１８】
前記解析モジュールは、
マクロを用いて可変パラメータリストを作成するための作成サブモジュールと、
前記可変パラメータリストに基づいて前記テンプレートパラメータを反復に解析するための解析サブモジュールと、を含み、
ここで、前記入力情報を前記マクロのマクロパラメータとする
請求項１７に記載の装置。
【請求項１９】
前記可変パラメータリストは、
処理対象演算子の名称と、
前記演算子カーネル関数の名称と、
前記演算子カーネル関数のデータレイアウトと、
前記深層学習フレームワークを実装するための機器タイプと、
前記深層学習フレームワークを実装するためのデータタイプと、を含む
請求項１８に記載の装置。
【請求項２０】
前記解析サブモジュールは、
前記マクロに含まれるパラメータ数を特定するための第１の特定ユニットと、
前記パラメータ数に応じて、反復終了位置を特定するための第２の特定ユニットと、
可変パラメータリストを用いて、前記反復終了位置に応じて前記入力情報を反復に処理するための処理ユニットと、を含む
請求項１８に記載の装置。
【請求項２１】
前記解析サブモジュールは、
前記可変パラメータリストを用いて前記テンプレートパラメータを反復に解析して、複数の内部テンプレートパラメータを得るための解析ユニットと、
前記複数の内部テンプレートパラメータをスプライシングして、前記複数の完全なテンプレートパラメータを得るためのスプライシングユニットと、をさらに含む
請求項１８～２０のいずれか一項に記載の装置。
【請求項２２】
前記複数の内部テンプレートパラメータは複数の文字列を含み、前記スプライシングユニットは、
前記入力情報に基づいて、前記複数の文字列をスプライシングして、前記処理対象演算子に対する複数の完全なテンプレートパラメータを得るためのスプライシングサブユニットを含む
請求項２１に記載の装置。
【請求項２３】
前記処理モジュールは、

前記複数の完全なテンプレートパラメータを明示的にインスタンス化することで、前記演算子カーネル関数を処理して、前記深層学習フレームワーク用の利用可能な演算子を得るための明示的なインスタンス化サブモジュールを含む
請求項２１に記載の装置。
【請求項２４】
前記複数の完全なテンプレートパラメータの各々は、テンプレート関数記述情報及びテンプレート関数情報を含み、前記明示的なインスタンス化サブモジュールは、
前記テンプレート関数記述情報をヘッダファイルとしてインスタンス化するための第１のインスタンス化ユニットと、
前記テンプレート関数情報をソースコードファイルとしてインスタンス化するための第２のインスタンス化ユニットと、を含む
請求項２３に記載の装置。
【請求項２５】
前記処理モジュールは、
前記複数の完全なテンプレートパラメータのうちの各完全なテンプレートパラメータに対して、該完全なテンプレートパラメータの関数パラメータタイプを特定するための第１の特定サブモジュールと、
前記関数パラメータタイプに基づいて、前記完全なテンプレートパラメータの入力情報を対応するパラメータ情報に変換するための変換サブモジュールと、
前記対応するパラメータ情報を記録するための記録サブモジュールと、を含む
請求項２１に記載の装置。
【請求項２６】
前記複数の完全なテンプレートパラメータに基づいて、前記深層学習フレームワークに基づく登録対象演算子を作成するための作成モジュールと、
前記登録対象演算子を前記深層学習フレームワーク内部のグローバル演算子テーブルに登録するための登録モジュールと、をさらに含む
請求項２１に記載の装置。
【請求項２７】
前記登録対象演算子は登録対象演算子記述情報及び登録対象演算子カーネル関数を含み、
前記登録対象演算子カーネル関数には正規化された入力情報及び正規化された関数ポインタが含まれ、前記作成モジュールは、
前記複数の完全なテンプレートパラメータのうちの各完全なテンプレートパラメータに対して、前記完全なテンプレートパラメータに対応する、正規化された形式を有する静的関数を含む構造体を特定するための第２の特定サブモジュールと、
前記静的関数の入力情報を前記正規化された入力情報とするための第１の正規化サブモジュールと、
前記静的関数の関数ポインタを前記正規化された関数ポインタとするための第２の正規化モジュールと、含む
請求項２６に記載の装置。
【請求項２８】
前記静的関数の入力情報は入力情報リストを構成し、前記入力情報リストは入力テンソルリストを含み、前記構造体は少なくとも１つの特化サブ構造体を含み、前記少なくとも１つの特化サブ構造体の各々は対応するデータタイプを有する
請求項２７に記載の装置。
【請求項２９】
前記グローバル演算子テーブルは、
演算子カーネル工場クラスと、
演算子カーネル名称クラス及びカーネルキー値クラスと、
演算子カーネルクラスと、を含む
請求項２６に記載の装置。

【請求項３０】
前記演算子カーネルクラスは、
演算子カーネル関数のポインタと、
演算子カーネル関数入力パラメータの記述情報と、
演算子カーネル関数出力パラメータの記述情報と、の少なくとも１つを含む
請求項２９に記載の装置。
【請求項３１】
前記深層学習フレームワークに依存しないテンプレートパラメータは、前記処理対象演算子を配布するために用いられ、前記深層学習フレームワークに依存しないテンプレートパラメータは、前記深層学習フレームワークの実装機器及び前記深層学習フレームワークのデータタイプのいずれにも依存しないテンプレートパラメータを含む
請求項１７に記載の装置。
【請求項３２】
前記利用可能な演算子に対する呼び出しコマンドに応答して、前記利用可能な演算子を呼び出すための呼び出しモジュールと、
呼び出された前記利用可能な演算子をコンパイルすることで、前記利用可能な演算子に対応する関数機能を実行するための実行モジュールと、をさらに含む
請求項１７に記載の装置。
【請求項３３】
少なくとも１つのプロセッサと、
前記少なくとも１つのプロセッサと通信接続するメモリと、を含み、
前記メモリに、前記少なくとも１つのプロセッサによって実行され得るコマンドが記憶されており、前記コマンドが前記少なくとも１つのプロセッサによって実行されることで、前記少なくとも１つのプロセッサが請求項１～１６のいずれか一項に記載の方法を実行することができる、
電子機器。
【請求項３４】
コンピュータに請求項１～１６のいずれか一項に記載の方法を実行させるためのコンピュータコマンドを記憶している、
非一時的なコンピュータ読取可能な記憶媒体。
【請求項３５】
プロセッサにより実行される場合に、請求項１～１６のいずれか一項に記載の方法を実現するコンピュータプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本願は、２０２２年５月１９日に提出した中国特許出願Ｎｏ．２０２２１０５６０８３１．９の優先権を要求し、その内容をまとめてここで参考にする。
【０００２】
＜技術分野＞
本開示はコンピュータ技術分野に関し、特に深層学習等の人工知能技術分野に関する。より具体的に、本開示は深層学習フレームワークの演算子処理方法、装置、電子機器、記憶媒体ならびにコンピュータプログラムを提供する。
【背景技術】
【０００３】
人工知能技術の発展に伴い、深層学習フレームワークはますます注目されてきている。深層学習フレームワークの評価指標はフレームワークの拡張性を含む。
【発明の概要】
【課題を解決するための手段】
【０００４】

本開示は、深層学習フレームワークの演算子処理方法及び装置、電子機器、記憶媒体ならびにコンピュータプログラムを提供する。

【０００５】

本開示の一態様によれば、深層学習フレームワークの演算子処理方法を提供し、該方法は、演算子カーネル関数、及び深層学習フレームワークに依存しないテンプレートパラメータを含む処理対象演算子を取得することと、処理対象演算子に対する入力情報を受信したことに応答して、入力情報を用いてテンプレートパラメータを解析して、深層学習フレームワークに依存する複数の完全なテンプレートパラメータを得ることと、複数の完全なテンプレートパラメータに基づいて、演算子カーネル関数を処理して、深層学習フレームワーク用の利用可能な演算子を得ることと、を含む。

【０００６】

本開示の別の態様によれば、深層学習フレームワークの演算子処理装置を提供し、該装置は、演算子カーネル関数、及び深層学習フレームワークに依存しないテンプレートパラメータを含む処理対象演算子を取得するための取得モジュールと、処理対象演算子に対する入力情報を受信したことに応答して、入力情報を用いてテンプレートパラメータを解析して、深層学習フレームワークに依存する複数の完全なテンプレートパラメータを得るための解析モジュールと、複数の完全なテンプレートパラメータに基づいて、演算子カーネル関数を処理して、深層学習フレームワーク用の利用可能な演算子を得るための処理モジュールと、を含む。

【０００７】

本開示の別の態様によれば、少なくとも１つのプロセッサと、該少なくとも１つのプロセッサと通信接続するメモリと、を含み、該メモリに、該少なくとも１つのプロセッサによって実行され得るコマンドが記憶されており、該コマンドが該少なくとも１つのプロセッサによって実行されることで、該少なくとも１つのプロセッサが本開示が提供した方法を実行することができる、電子機器を提供した。

【０００８】

本開示の別の態様によれば、コンピュータに本開示が提供した方法を実行させるためのコンピュータコマンドを記憶している、非一時的なコンピュータ読取可能な記憶媒体を提供した。

【０００９】

本開示の別の態様によれば、プロセッサにより実行される場合に、本開示が提供した方法を実現するコンピュータプログラムを提供した。

【００１０】

理解されるべきこととして、本部分に記載された内容は、本開示の実施例のキーポイント又は重要な特徴を示すことを意図するものではなく、本開示の範囲を限定するものでもない。本開示の他の特徴は、以下の説明により容易に理解される。

【００１１】

ここで、図面は、本技術案をよりよく理解するために用いられ、本開示を限定するものではない。

【図面の簡単な説明】

【００１２】

【図１】本開示の一実施例による深層学習フレームワークの演算子処理方法のフローチャートである。

【００１３】

【図２】本開示の一実施例による深層学習フレームワークの演算子処理方法のフローチャートである。

【００１４】

【図３】本開示の一実施例による深層学習フレームワークの演算子処理方法のフローチャートである。

【００１５】

【図４】本開示の一実施例による深層学習フレームワークの演算子処理方法のフローチャートである。
【００１６】
【図５】本開示の一実施例による深層学習フレームワークの演算子処理方法のフローチャートである。
【００１７】
【図６】本開示の一実施例によるグローバル演算子テーブルの例示的模式図である。
【００１８】
【図７】本開示の一実施例による深層学習フレームワークの演算子処理装置のブロック図である。
【００１９】
【図８】本開示の一実施例による深層学習フレームワークの演算子処理方法を適用可能な電子機器のブロック図である。
【発明を実施するための形態】
【００２０】
以下、図面を参照して本開示の例示的な実施例を説明する。ここで、より理解しやすいために本開示の実施例の様々な詳細は含まれ、それらが例示的なものであると考えられるべきである。したがって、当業者であれば、ここで記載される実施例に対して様々な変更・修正を行うことができ、本開示の範囲及び精神から逸脱することはないと分るべきである。同様に、明確かつ簡潔に説明するために、以下の記載において周知の機能や構成に対する説明を省略する。
【００２１】
深層学習フレームワークは、人工知能分野のインフラストラクチャであり、より成熟した発展段階に入った。上層拡張および二次開発をより効率的かつ便利的にサポートすることができるかどうかは、深層学習フレームワーク製品が「インフラストラクチャ」として機能することができるかどうか、人工知能産業の着地を実現することができるかどうか、および分野拡張を行うことができるかどうかを評価する重要な指標である。
【００２２】
深層学習フレームワークは、ＰａｄｄｌｅＰａｄｄｌｅ（パドル）フレームワーク、ＰｙＴｏｒｃｈフレームワーク、ＴｅｎｓｏｒＦｌｏｗフレームワーク、ＭｉｎｄＳｐｏｒｅフレームワーク及び他の複数のニッチフレームワークを含む。これらのフレームワークは、類似の機能を提供することができる。しかしながら、人工知能時代の生態建設の「基石」として、深層学習フレームワークは、フレームワークがさらに加工されて改造されるように、より多くの上流開発者を吸引する必要がある。そのため、深層学習フレームワークは、上流開発者がフレームワークの生態を共同構築するコストを低減するために、設計が明確で、メンテナンスしやすく、理解しやすく、カスタマイズ化拡張開発しやすいなどの特徴を有する必要がある。演算子体系は、１つのフレームワークの半数以上のコードを占めることができ、深層学習フレームワークの主体とすることができる。低コストで演算子体系を拡張して追加することができるか否かは、そのうちのフレームワークの生態を共同構築するコストを低減する重要な一環である。
【００２３】
演算子は、深層学習フレームワークにおけるテンソル計算ユニットである。演算子は、特定のテンソル組み合わせを入力とし、特定の計算ロジックを完成し、計算結果を返すことができる。深層学習フレームワークを用いてモデルを構築するプロセスは、異なる演算子を組み合わせて特定の計算ロジックを満たすプロセスであってもよい。
【００２４】
演算子の多様性と豊富さは、深層学習フレームワークを評価する重要な指標とすることができる。深層学習フレームワークにおける演算子は、継続的に動的に追加されてもよい。異なる演算子の間には、関連もあるし、区別もある。複雑な演算子は、簡単な演算子の組み合わせによって実装されることが多い。深層学習フレームワークがこのような演算子の

組み合わせによる実装のモードをサポートすると、コスト削減及び効率アップに寄与し、外部開発者を、該フレームワークを二次開発するように吸引することにも寄与する。

【００２５】

関数は、プログラミング言語における基礎パラダイムである。関数同士は互いに呼び出されてもよい。深層学習フレームワークに便利な演算子組合せ開発能力を持たせるために、関数式演算子体系に基づいて深層学習フレームワークを構築することができる。

【００２６】

関数式演算子体系の応用が少ない。関数と演算子との間には矛盾がある。例えば、関数自体が簡単すぎて、情報と特徴を携帯する能力がない。また、関数が柔軟になりすぎ、形式が多様であり、パラメータが多様であり、正規化が困難である。演算子は、比較的複雑な概念であり、特徴記述を有し、様々な機器に適合する必要がある。また、演算子は、フレームワークの統合スケジューリングユニットとして、グローバルスケジューリングに適合するように、一致する形態を抽象化する必要がある。

【００２７】

現実的に考慮すると、深層学習フレームワーク製品は、トレードオフの設計を行うことが多く、関数式のパラダイムを放棄し、「構造体+内部計算方法」の形式を使用し、組み合わせ多重化の利便性を犠牲にして、演算子のパラダイムの統一を保持する。しかしながら、このようなトレードオフ設計は、フレームメンテナンスコスト及び拡張コストの増大を招く。

【００２８】

いくつかの深層学習フレームワーク製品は、演算子の多様性を遮蔽するために、トップレベル演算子関数をカプセル化することができる。これらの深層学習フレームワーク製品は、関数内部で配布選択を行う方式を利用して、演算子同士の多重化を実現することができる。演算子の間の多重化時に、これらの深層学習フレームワーク製品は、配布選択のスケジューリングオーバーヘッドが発生し、性能が損なわれる。これらの深層学習フレームワーク製品は、演算子の組み合わせ多重化の利便性を保有するが、一部の性能を犠牲にする。演算子の組み合わせを行わなければ、性能をより良くすることができ、演算子の多重化方式を放棄し、複数の演算子カーネルのコードを直接にまとめて、性能を向上させることができる。

【００２９】

したがって、関数式演算子同士の多重化性能の問題を解決すれば、関数式演算子の優位性を永続的に発揮させることができる。

【００３０】

「関数式」と「高性能多重化」を両立する演算子体系を形成するために、形式が簡単で、柔軟で多様な「関数」を形式が統一され且つ記述特徴を有する演算子として自動的にパッケージし、「関数式」の開発を実現する必要があり、それにより関数式演算子の多重化時に追加のスケジューリング配布オーバーヘッドを引き込まないことを確保し、「高性能多重化」を実現する。

【００３１】

関数式演算子体系を実装するために、関数式演算子のパラダイム設計と関数式演算子の解析及び登録が必要である。関数式演算子のパラダイム設計は、関数の書き方により演算子の計算カーネルを実装することである。関数において、一般的にＣ＋＋プログラミング言語を用いて演算子のコア計算ロジックを作成する。関数式演算子は、自然に演算子同士の相互多重化をサポートする。関数式演算子の解析及び登録は、キーコンポーネントを設計し、形式が簡単で、柔軟で多様な関数を形式が統一され且つ記述特徴を有する演算子として自動的にパッケージし、フレームワークの統合スケジューリングに供する。

【００３２】

また、多重化可能な関数式演算子体系を実装するために、関数式演算子同士が機器非依存の多重化をサポートする必要もある。同一の演算子が異なる機器に適用される場合、異なるカーネル関数が必要となる。例えば、演算子がＣＰＵ（Ｃｅｎｔｒａｌ　Ｐｒｏｃｅｓ

ｓｉｎｇ Ｕｎｉｔ，中央処理装置）に適用される場合、ＣＰＵの演算子カーネルが必要であり、ＧＰＵ（Ｇｒａｐｈｉｃｓ Ｐｒｏｃｅｓｓｉｎｇ Ｕｎｉｔ，グラフィックプロセッサ）がＧＰＵの演算子カーネルが必要であり、関数式演算子であっても、この２つのカーネルは異なる関数である必要があり、１つの演算子を新たに開発し、それが他の演算子を多重化して組み合わせて実装することができる場合、多重化される演算子は機器非依存である必要があり、そうでない場合、重複開発を招き、簡単な例を挙げる。

【００３３】

２つの演算子が既に存在すると仮定する。

【００３４】

- 演算子Ａ：
    - 演算子Ａ ＣＰＵカーネル：A_CPU()
    - 演算子Ａ ＧＰＵカーネル：A_GPU()

【００３５】

- 演算子Ｂ：
    - 演算子Ｂ ＣＰＵカーネル：B_CPU()
    - 演算子Ｂ ＧＰＵカーネル：B_GPU()

【００３６】

例えば、演算子Ａ及び演算子Ｂに基づいて１つの新たな演算子を実装する。

【００３７】

１つの実装方式は、以下の通りである。

【００３８】

- 演算子Ｃ（演算子ＡとＢを多重化して実装する）：
    - 演算子Ｃ ＣＰＵカーネル：C_CPU(){ A_CPU();B_CPU(); }
    - 演算子Ｃ ＧＰＵカーネル：C_GPU(){ A_GPU();B_GPU(); }

【００３９】

該実装方式は合理的な関数式演算子体系ではなく、冗長コードが多く、メンテナンスコストが高い。また、このような方式による多重化演算子は、機器分けで多重化することしかできず、各機器の演算子Ｃカーネルの実装コードがほぼ同じであることを容易に発見することができる。関数式演算子体系は、演算子の多重化効率を向上させ、メンテナンスコストを低減し、拡張開発を容易にするためのものである。このような各機器カーネルは、１つのカーネルをほぼコピーして実装する必要があり、厳密な関数式演算子体系ではない。

【００４０】

上述したように、異なる深層学習フレームワークは、例えばＰｙｔｏｒｃｈフレームワーク、ＴｅｎｓｏｒＦｌｏｗフレームワーク、及びＭｉｎｄＳｐｏｒｅフレームワークを含んでもよい。

【００４１】

Ｐｙｔｏｒｃｈフレームワークでは、演算子は最終的に１つの関数式のインターフェースにカプセル化され、関数式演算子同士は多重化されてもよく、Ｐｙｔｏｒｃｈの１つの演算子は複数の層に分けて実装され、上層が下層を呼び出す場合、次の層の演算子を検索して実装する必要がある。ＰｙＴｏｒｃｈフレームワークでは、演算子同士の多重化、及び演算子内の多重化の場合、演算子カーネルマッピングテーブルを用いて対応するカーネルを検索して選択する方式を採用する。このような方式のオーバーヘッドは比較的大きく、１つの演算子の実行過程において、複数回の演算子検索を行う必要がある可能性がある。

【００４２】

ＴｅｎｓｏｒＦｌｏｗフレームワークでは、演算子は依然として構造体演算子であり、関数式演算子ではない。構造体形式の演算子カーネルの実装は、直感的ではなく、追加の概念を増加し、且つ演算子同士の多重化には、より複雑な書き方及びより大きいスケジューリングオーバーヘッドを引き込む必要があり、設計上で、新たな演算子の組み合わせ開発に不便な形式に自然に属する。

【００４３】

一般的に、構造体形式の演算子体系では、演算子カーネル関数を実装する際に、異なる演算子が当該演算子カーネル関数を多重化できるように、関数を別途定義する必要がある。このような方式は、ある程度の一般化を実現することができるが、メンテナンスコストが高く、規範的管理が困難であり、フレームワークの長期発展により多くの冗長なコードが現れることになり、メンテナンスコストが徐々に増大する。

【００４４】

ＴｅｎｓｏｒＦｌｏｗフレームワークの派生製品ｔｅｎｓｏｒｆｌｏｗ／ｒｕｎｔｉｍｅフレームワークにおいて、演算子は関数式演算子であってもよい。しかし、ｔｅｎｓｏｒｆｌｏｗ／ｒｕｎｔｉｍｅフレームワークは１つの実験的製品に過ぎず、演算子の数が少なく、まだ体系になっていない。また、演算子パラメータリストが混乱し、規範性が悪く、高性能な多重化を実現できない。

【００４５】

ＭｉｎｄＳｐｏｒｅフレームワークでの演算子は、ＴｅｎｓｏｒＦｌｏｗフレームワークと類似し、説明を繰り返さない。

【００４６】

図１は、本開示の一実施例による深層学習フレームワークの演算子処理方法のフローチャートである。

【００４７】

図１に示すように、該方法１００は、操作Ｓ１１０～操作Ｓ１３０を含んでもよい。

【００４８】

操作Ｓ１１０において、処理対象演算子を取得する。

【００４９】

例えば、処理対象演算子は、演算子カーネル関数、及び深層学習フレームワークに依存しないテンプレートパラメータを含む。

【００５０】

例えば、取得された処理対象演算子は、例えばＳｃａｌｅ（スケーリング）演算子であってもよい。Ｓｃａｌｅ演算子は、演算子カーネル関数ｓｃａｌｅ_ｋｅｒｎｅｌ（スケーリング演算子カーネル関数）及びテンプレートパラメータＴを含んでもよい。ｓｃａｌｅ_ｋｅｒｎｅｌは、ｓｃａｌｅ演算を行うことができる。

【００５１】

例えば、深層学習フレームワークに依存しないテンプレートパラメータは、例えば深層学習フレームワークの実装機器に依存しないテンプレートパラメータを含んでもよく、深層学習フレームワークのデータタイプに依存しないテンプレートパラメータを含んでもよい。

【００５２】

例えば、実装機器は、例えばＣＰＵ又はＧＰＵであってもよい。また例えば、データタイプは、例えばＦｌｏａｔ（浮動小数点）タイプ、Ｉｎｔ（整数）タイプであってもよい。

【００５３】

例えば、Ｓｃａｌｅ演算子を処理対象演算子とする場合、処理対象演算子は、例えばＳｃａｌｅＫｅｒｎｅｌ〈Ｔ、 Ｃｏｎｔｅｘｔ〉()と表すことができる。

【００５４】

操作Ｓ１２０において、処理対象演算子に対する入力情報を受信したことに応じて、入力情報を用いてテンプレートパラメータを解析して、深層学習フレームワークに依存する複数の完全なテンプレートパラメータを得る。

【００５５】

例えば、１つの完全なテンプレートパラメータは、〈Ｆｌｏａｔ、 ＣＰＵＣｏｎｔｅｘｔ〉であってもよい。Ｆｌｏａｔはデータタイプであり、ＣＰＵＣｏｎｔｅｘｔは機器タイプである。

【００５６】

操作Ｓ１３０において、複数の完全なテンプレートパラメータに基づいて、演算子カーネル関数を処理して、深層学習フレームワーク用の利用可能な演算子を得る。

【００５７】
例えば、処理対象演算子を取得した後、解析により具体的な深層学習フレームワークの利用可能な演算子を得て、該深層学習フレームワークに該利用可能な演算子を登録することで、深層学習フレームワークにおける他のルーチン又はタスクが、実際の深層学習アプリケーションに応じて該利用可能な演算子を呼び出して対応するアルゴリズム又は機能、例えばスケーリング、畳み込み等を実行することができる。
【００５８】
例えば、完全なテンプレートパラメータ〈Ｆｌｏａｔ、 ＣＰＵＣｏｎｔｅｘｔ〉に基づいて、演算子カーネル関数ｓｃａｌｅ_ｋｅｒｎｅｌを処理することで、利用可能な演算子を得ることができる。利用可能な演算子は、例えばＳｃａｌｅＫｅｒｎｅｌ〈Ｆｌｏａｔ、ＣＰＵＣｏｎｔｅｘｔ〉()と表すことができる。
【００５９】
本開示の実施例によれば、演算子が多重化される時、スケジューリングオーバーヘッドがないことを実現し、高性能多重化を実現することができる。
【００６０】
幾つかの実施例において、方法１００は、利用可能な演算子に対する呼び出しコマンドに応答して、利用可能な演算子を呼び出すことと、呼び出された利用可能な演算子をコンパイルすることで、利用可能な演算子に対応する関数機能を実行することとをさらに含んでもよい。
【００６１】
例えば、利用可能な演算子ＳｃａｌｅＫｅｒｎｅｌ〈ｆｌｏａｔ、 ＣＰＵＣｏｎｔｅｘｔ〉()に対する呼び出しコマンドに応答して、利用可能な演算子ＳｃａｌｅＫｅｒｎｅｌ〈ｆｌｏａｔ、 ＣＰＵＣｏｎｔｅｘｔ〉()を呼び出し、呼び出された利用可能な演算子ＳｃａｌｅＫｅｒｎｅｌ〈ｆｌｏａｔ、 ＣＰＵＣｏｎｔｅｘｔ〉()をコンパイルすることで、ｓｃａｌｅ関数の機能を実行する。
【００６２】
幾つかの実施例において、深層学習フレームワークの実装機器に依存しない演算子カーネル関数の宣言は、例えば以下の通りであってもよい。
【００６３】
【数１】

```
template <typename T, typename Context>
void ScaleKernel(const Context& dev_ctx,
                 const DenseTensor& x,
                 const Scalar& scale,
                 float bias,
                 bool bias_after_scale,
                 DenseTensor* out);
```

【００６４】
Ｔｅｍｌａｔｅはテンプレートである。ｔｙｐｅｎａｍｅ Ｔは、深層学習フレームワークのデータタイプに依存しないテンプレートパラメータである。ｔｙｐｅｎａｍｅ Ｃｏｎｔｅｘｔは、深層学習フレームワークの実装機器に依存しないテンプレートパラメータである。ｔｙｐｅｎａｍｅ Ｔ及びｔｙｐｅｎａｍｅ Ｃｏｎｔｅｘｔは、具体的な実装機器にバインドされなくてもよい。
【００６５】

また例えば、基本演算を多重化可能な演算子Ｃａｌｃは、例えば以下の通りであってもよい。

【００６６】

【数２】

```
template <typename T, typename Context>
void CalcKernel(const Context& dev_ctx, ...) {
    ScaleKernel<T, Context>(...)
}
```

【００６７】

演算子Ｃａｌｃが呼び出した演算子は、演算子ＳｃａｌｅＫｅｒｎｅｌ〈Ｔ、 Ｃｏｎｔｅｘｔ〉()である。演算子ＳｃａｌｅＫｅｒｎｅｌ〈Ｔ、 Ｃｏｎｔｅｘｔ〉()は、実装機器に依存しない演算子である。コンパイル時に、テンプレートパラメータＴ及びＣｏｎｔｅｘｔを取得し、コンパイラにより具体的なデータタイプ及び実装機器に変換する。一例において、演算子ＳｃａｌｅＫｅｒｎｅｌ〈Ｔ、 Ｃｏｎｔｅｘｔ〉()がＳｃａｌｅＫｅｒｎｅｌ〈ｆｌｏａｔ、 ＣＰＵＣｏｎｔｅｘｔ〉()に変換されてもよい。実行時に、演算子Ｃａｌｃは、ｓｃａｌｅ演算のｆｌｏａｔタイプを呼び出し、ＣＰＵ機器のコマンドセットに基づいて、他のロジックを必要とせずに判断及び選択を行い、スケジューリングオーバーヘッド無しの多重化ができて、高性能多重化を実現することができる。

【００６８】

幾つかの実施例において、処理対象演算子に対する入力情報を受信したことに応じて、入力情報を用いてテンプレートパラメータを解析して、深層学習フレームワークに依存する複数の完全なテンプレートパラメータを得ることは、マクロを用いて可変パラメータリストを作成することと、可変パラメータリストに基づいてテンプレートパラメータを反復的に解析することとを含み、ここで、入力情報をマクロのマクロパラメータとする。

【００６９】

例えば、マクロは、Ｃ言語のメカニズムの一種であり、重複のコードを簡略化することに用いられ、コンパイル前処理時に、該マクロを対応するコードセグメントに置換えて実行する。

【００７０】

例えば、上述した処理対象演算子ＳｃａｌｅＫｅｒｎｅｌ〈Ｔ、 Ｃｏｎｔｅｘｔ〉()に対して、入力情報に基づいて、可変パラメータリストを特定してもよい。マクロにより可変パラメータリストに基づいて反復的に解析する。一例において、マクロＰＤ＿ＲＥＧＩＳＴＥＲ＿ＫＥＲＮＥＬは、可変パラメータリストを解析することができる。

【００７１】

【数３】

```
PD_REGISTER_KERNEL(scale,
                   CPU,
                   ALL_LAYOUT,
                   ScaleKernel,
                   float,
                   double,
                   bfloat16,
                   uint8_t,
                   int8_t,
                   int16_t,
                   int,
                   int64_t) {}
```

【００７２】
幾つかの実施例において、可変パラメータリストは、処理対象演算子の名称と、演算子カーネル関数の名称と、演算子カーネル関数のデータレイアウトと、深層学習フレームワークを実装するための機器タイプと、深層学習フレームワークを実装するためのデータタイプとを含む。
【００７３】
例えば、上述したＳｃａｌｅ演算子に対して、マクロを用いて作成されたパラメータリストにおいて、ｓｃａｌｅは演算子名称であり、ＣＰＵは深層学習フレームワークを実装するための機器タイプであり、ＡＬＬ＿ＬＡＹＯＵＴは演算子カーネル関数ＳｃａｌｅＫｅｒｎｅｌがすべてのデータレイアウトに適合し得ることを表し、ＳｃａｌｅＫｅｒｎｅｌは登録及び解析しようとする演算子カーネル関数名を表し、ｆｌｏａｔ、ｄｏｕｂｌｅ、ｂｆｌｏａｔ１６、ｕｉｎｔ８＿ｔ、ｉｎｔ１６＿ｔ、ｉｎｔ、ｉｎｔ６４＿ｔは深層学習フレームワークがサポートするデータタイプを表す。
【００７４】
幾つかの実施例において、可変パラメータリストに基づいてテンプレートパラメータを反復的に解析することは、マクロに含まれるパラメータ数を特定することと、パラメータ数に応じて、反復終了位置を特定することと、可変パラメータリストを用いて、反復終了位置に応じて入力情報を反復的に処理することとを含む。
【００７５】
例えば、上述した処理対象演算子ＳｃａｌｅＫｅｒｎｅｌ〈Ｔ、 Ｃｏｎｔｅｘｔ〉()に対して、マクロＰＤ＿ＲＥＧＩＳＴＥＲ＿ＫＥＲＮＥＬを用いてパラメータリストを解析する時に、別のマクロＰＤ＿ＮＡＲＧＳを用いてマクロＰＤ＿ＲＥＧＩＳＴＥＲ＿ＫＥＲＮＥＬに含まれるパラメータ数を特定してもよい。
【００７６】
また例えば、パラメータ数を特定した後、パラメータ数をマクロの添え字とすることで、下層マクロ方法をスプライシングし、その後、該マクロ方法を呼び出して解析することができる。
【００７７】
例えば、マクロＰＤ＿ＮＡＲＧＳは、例えば以下の通りであってもよい。

【００７８】
【数４】

```
#define PD_NARGS(...) _PD_NARGS((__VA_ARGS__, _PD_RESQ_N()))

#define _PD_NARGS(...) _PD_ARG_N(__VA_ARGS__)

#define _PD_ARG_N_EXPAND(                                    \
    _1, _2, _3, _4, _5, _6, _7, _8, _9, _10, _11, _12, _13, _14, _15, N, ...) \
    N

#define _PD_ARG_N(args) _PD_ARG_N_EXPAND args

#define _PD_RESQ_N() 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
```

【００７９】
例えば、ＰＤ_ＮＡＲＧＳ(...)に対して、「...」は上述した可変パラメータリストであってもよい。本開示の実施例によれば、マクロに幾つのパラメータが存在するかを特定し、対応するパラメータ数の位置からマクロパラメータを反復的に解析することで、反復終了位置を特定することができる。
【００８０】
他の実施例において、マクロＰＤ_ＮＡＲＧＳが関連処理を行うために、マクロＰＤ_ＮＡＲＧＳを可変パラメータマクロ_ＶＡ_ＡＲＧＳの外にネストしてもよい。
【００８１】
例えば、以下の方式によってマクロＰＤ_ＮＡＲＧＳを可変パラメータマクロＶＡ_ＡＲＧＳの外にネストすることができる。
【００８２】
【数５】

```
_PD_KERNEL_REGISTRAR_INIT(PD_NARGS(__VA_ARGS__), \
                                    reg_type,                   \
                                    kernel_name,                \
                                    backend,                    \
                                    context,                    \
                                    layout,                     \
                                    args_def_fn,                \
```

【００８３】
反復終了位置が特定された後、可変パラメータリストに基づいてテンプレートパラメータを反復的に解析することができる。
【００８４】
幾つかの実施例において、可変パラメータリストに基づいてテンプレートパラメータを反復的に解析することは、可変パラメータリストを用いて、入力情報に基づいてテンプレートパラメータを反復的に解析して、複数の内部テンプレートパラメータを得ることと、複数の内部テンプレートパラメータをスプライシングして、複数の完全なテンプレートパラメータを得ることとをさらに含んでもよい。以下、図２を参照して詳細に説明する。

【００８５】
図２は、本開示の別の実施例による深層学習フレームワークの演算子処理方法のフローチャートである。
【００８６】
図２に示すように、方法２２１は、可変パラメータリストに基づいてテンプレートパラメータを反復的に解析する。以下、操作Ｓ２２１１及び操作Ｓ２２１２を参照して詳細に説明する。
【００８７】
操作Ｓ２２１１において、可変パラメータリストを用いてテンプレートパラメータを反復的に解析して、複数の内部テンプレートパラメータを得る。
【００８８】
例えば、マクロＶＡ_ＡＲＧＳを用いてマクロパラメータを反復することで、可変パラメータリストに基づいてテンプレートパラメータを反復的に解析する。
【００８９】
例えば、可変パラメータリストを用いて、以下の方式によって反復的に解析することで、テンプレートパラメータを得ることができる。
【００９０】
【数６】

```
#define _PD_KERNEL_REGISTRAR_INIT_3(cpp_dtype, ...)

#define _PD_KERNEL_REGISTRAR_INIT_4(...)

    _PD_KERNEL_REGISTRAR_INIT_3(__VA_ARGS__)
```

【００９１】
マクロＰＤ_ＫＥＲＮＥＬ_ＲＥＧＩＳＴＲＡＲ_ＩＮＩＴ_４に対して、そのマクロパラメータ「…」は、上述した可変パラメータリストである。一例において、マクロパラメータは、例えばｆｌｏａｔ、ｄｏｕｂｌｅ、ｂｆｌｏａｔ１６、ｕｉｎｔ８_ｔ、ｉｎｔ８_ｔ、ｉｎｔ１６_ｔ、ｉｎｔ、ｉｎｔ６４_ｔを含む。
【００９２】
マクロＰＤ_ＫＥＲＮＥＬ_ＲＥＧＩＳＴＲＡＲ_ＩＮＩＴ_４の内部は、マクロＰＤ_ＫＥＲＮＥＬ_ＲＥＧＩＳＴＲＡＲ_ＩＮＩＴ_３を呼び出すことができる。
【００９３】
マクロＰＤ_ＫＥＲＮＥＬ_ＲＥＧＩＳＴＲＡＲ_ＩＮＩＴ_３は、マクロＶＡ_ＡＲＧＳの外にネストして、可変パラメータリストを全体として反復するために用いられる。マクロＰＤ_ＫＥＲＮＥＬ_ＲＥＧＩＳＴＲＡＲ_ＩＮＩＴ_３は、可変パラメータリストにおける最初のパラメータを解析し、例えばｆｌｏａｔを解析することができる。この場合、ｃｐｐ_ｄｔｙｐｅはｆｌｏａｔであってもよく、マクロＰＤ_ＫＥＲＮＥＬ_ＲＥＧＩＳＴＲＡＲ_ＩＮＩＴ_４ のマクロパラメータ「…」には、ｄｏｕｂｌｅ、ｂｆｌｏａｔ１６、ｕｉｎｔ８_ｔ、ｉｎｔ８_ｔ、ｉｎｔ１６_ｔ、ｉｎｔ、ｉｎｔ６４_ｔが含まれてもよい。次回の反復時に、ｄｏｕｂｌｅは可変パラメータリストにおける最初のパラメータになる。
【００９４】
操作Ｓ２２１２において、複数の内部テンプレートパラメータをスプライシングして、複数の完全なテンプレートパラメータを得る。
【００９５】
本開示の実施例において、複数の内部テンプレートパラメータは複数の文字列を含む。
【００９６】
本開示の実施例において、複数の内部テンプレートパラメータをスプライシングして、複数の完全なテンプレートパラメータを得ることは、入力情報に基づいて、複数の文字列をスプライシングして、処理対象演算子に対する複数の完全なテンプレートパラメータを得

ることを含む。
【００９７】
例えば、解析された内部テンプレートパラメータは文字列である。複数の文字列は、例えば文字列Ｆｌｏａｔ及び文字列ＣＰＵＣｏｎｔｅｘｔを含む。この２つの文字列をスプライシングして、完全なテンプレートパラメータ〈Ｆｌｏａｔ、 ＣＰＵＣｏｎｔｅｘｔ〉を得ることができる。
【００９８】
幾つかの実施例において、複数の完全なテンプレートパラメータに基づいて、演算子カーネル関数を処理して、深層学習フレームワーク用の利用可能な演算子を得ることは、複数の完全なテンプレートパラメータのうちの各完全なテンプレートパラメータについて、該完全なテンプレートパラメータの関数パラメータタイプを特定することと、関数パラメータタイプに基づいて、完全なテンプレートパラメータの入力情報を対応するパラメータ情報に変換することと、対応するパラメータ情報を記録することとを含む。以下、図３を参照して詳細に説明する。
【００９９】
図３は、本開示の一実施例による深層学習フレームワークの演算子処理方法のフローチャートである。
【０１００】
図３に示すように、該方法３３０は、例えばＩ個の完全なテンプレートパラメータをトラバースし、第ｉ個の完全なテンプレートパラメータに対して操作Ｓ３３１～操作Ｓ３３５を含む操作を実行することで、完全なテンプレートパラメータの入力情報を対応するパラメータ情報に変換する。Ｉは１以上の整数であり、ｉはＩ以下の整数である。
【０１０１】
操作Ｓ３３１において、第ｉ個の完全なテンプレートパラメータの関数パラメータタイプを特定する。
【０１０２】
例えば、第ｉ個の完全なテンプレートパラメータに基づいて、幾つかの関数パラメータのタイプを関数パラメータタイプとして特定することができる。
【０１０３】
操作Ｓ３３２において、関数パラメータタイプに基づいて、第ｉ個の完全なテンプレートパラメータの入力情報を第ｉ組のパラメータ情報に変換する。
【０１０４】
例えば、異なる関数パラメータタイプは、異なる変換方式に対応していてもよい。関数パラメータタイプと変換方式との間の対応関係は予め設定されてもよい。
【０１０５】
また例えば、上述した処理対象演算子に対する入力情報には、第ｉ個の完全なテンプレートパラメータに対応する第ｉ個の入力情報が含まれてもよい。第ｉ個の入力情報には、複数の関数パラメータが含まれてもよい。対応する関数パラメータタイプに基づいて、入力情報を対応する第ｉ組のパラメータ情報に変換することができる。
【０１０６】
操作Ｓ３３３において、第ｉ組のパラメータ情報を記録する。
【０１０７】
例えば、第ｉ組のパラメータ情報をメモリに記録してもよい。
【０１０８】
操作Ｓ３３４において、ｉがＩに等しいか否かを判定する。
【０１０９】
ｉ＝Ｉの場合、操作Ｓ３３５を実行し、フローを終了する。ｉ＝Ｉの場合、既にＩ個の完全なテンプレートパラメータがトラバースされ、フローを終了することができる。
【０１１０】
ｉがＩより小さい場合、操作Ｓ３３１に戻り、第ｉ＋１個の完全なテンプレートパラメータ

に対して操作Ｓ３３１～操作Ｓ３３４を実行する。
【０１１１】
一例において、データタイプが８種類であり、且つ機器種別が２種類である場合、反復トラバースの方式によって、１６組のパラメータ情報を得て、異なる機器タイプ及びデータタイプの組み合わせをサポートすることができる。
【０１１２】
一例において、上述した第ｉ個の関数の入力情報に対して、第ｉ個の関数の関数パラメータタイプを特定することができ、そのタイプ及び関連する必要な情報を記録する。例えば、Ｃ＋＋テンプレートメタプログラミング技術を用いることで、コンパイル期間の計算、判断、変換、照会等の機能を実現する。
【０１１３】
幾つかの実施例において、例えば上述した方法３００は、複数の完全なテンプレートパラメータに基づいて、深層学習フレームワークに基づく登録対象演算子を作成することと、登録対象演算子を深層学習フレームワーク内部のグローバル演算子テーブルに登録することとをさらに含んでもよい。
【０１１４】
深層学習フレームワーク内部において、統合スケジューリングを実現するために、深層学習フレームワーク内部の演算子に対応する関数書き方は正規化されもよく、且つ該正規化された書き方とユーザのカスタマイズ演算子の計算関数の作成方式とは大きく異なる。
【０１１５】
例えば、異なるシーンにおいて、異なるユーザの計算関数の書き方は相違する可能性があり、例えば１つのテンソルを入力とする場合、関数の書き方は以下の通りである。
【０１１６】
Ｔｅｎｓｏｒリスト演算子計算関数名(入力Ｔｅｎｓｏｒ１){ ...}を返す。
【０１１７】
一方、２つの入力テンソルがあれば、関数の書き方は、以下の通りである。
【０１１８】
Ｔｅｎｓｏｒリスト演算子計算関数名(入力Ｔｅｎｓｏｒ１，入力Ｔｅｎｓｏｒ２){ ...}を返す。
【０１１９】
より多くの入力がある場合、より多くの関数宣言もある。Ｃ＋＋プログラミング言語において、異なる関数宣言は、異なる関数ポインタタイプ、つまり、異なるデータタイプを意味し、フレームワークはユーザが作成し得る関数ポインタタイプを記憶する必要があり、フレームワークの最下層で呼び出すことができ、このような書き方の柔軟性が極めて高いパラダイムは、Ｃ＋＋プログラミング言語において、簡潔で汎用的な形式で記憶することができない。全体の簡潔を実現するために、ユーザが作成した様々な計算関数を正規化する必要もある。
【０１２０】
例えば、１つのカスタマイズ計算関数の関数形式は以下の通りである。
【０１２１】
Ｔｅｎｓｏｒリスト演算子計算関数名(入力Ｔｅｎｓｏｒ１){ ...}を返す。
【０１２２】
別のカスタマイズ計算関数の関数形式は以下の通りである。
【０１２３】
Ｔｅｎｓｏｒリスト演算子計算関数名(入力Ｔｅｎｓｏｒ１，入力Ｔｅｎｓｏｒ２){ ...} を返す。
【０１２４】
上記２つのカスタマイズ計算関数の関数形式が統一されておらず、本実施例において、深層学習フレームワーク内部で統合スケジューリングを容易にするために、異なるカスタマイズ計算関数を正規化する必要がある。

【０１２５】
本開示の実施例において、登録対象演算子は、登録対象演算子記述情報及び登録対象演算子カーネル関数を含み、登録対象演算子カーネル関数は、正規化された入力情報及び正規化された関数ポインタを含み、複数の完全な内部テンプレート関数に基づいて、深層学習フレームワークに基づく登録対象演算子を作成することは、複数の完全なテンプレートパラメータのうちの各完全なテンプレートパラメータに対して、完全なテンプレートパラメータに対応する、正規化された形式を有する静的関数を含む構造体を特定することと、静的関数の入力情報を正規化された入力情報することと、静的関数の関数ポインタを正規化された関数ポインタとすることとを含む。
【０１２６】
例えば、深層学習フレームワーク内の各種の演算子は、ＯｐＫｅｒｎｅｌクラスを継承し、自身の計算関数（Ｃｏｍｐｕｔｅ）を充填して実装することができる。演算子の対応する入出力テンソルは、実行コンテキスト（ＥｘｅｃｕｔｉｏｎＣｏｎｔｅｘｔ）に記憶されてもよく、実行コンテキストのＩｎｐｕｔ及びＯｕｔｐｕｔ機能によって取得することができる。一例において、登録対象演算子カーネル関数を構築する時、正規化された形式の関数に基づいて構築することができ、該正規化された関数は、完全なテンプレートパラメータに対応する構造体における静的関数であってもよい。
【０１２７】
また例えば、登録対象演算子カーネル関数を構築する時、上記静的関数の入力情報を作成することと、上記静的関数の関数ポインタを作成することと、他の標準処理、例えば、ｃｏｎｔｅｘｔから入力情報を取得してｉｎｐｕｔｓ変数に入れる処理を作成することとを含む。
【０１２８】
登録対象演算子カーネル関数を構築する時に統一された形式の静的関数を採用することで、深層学習フレームワークの統合スケジューリングのために、カスタマイズ計算関数を統一することができる。
【０１２９】
さらに、幾つかの実施例において、静的関数の入力情報は入力情報リストを構成し、入力情報リストは入力テンソルリストを含み、構造体は少なくとも１つの特化サブ構造体を含み、少なくとも１つの特化サブ構造体の各々は対応するデータタイプを有し、データタイプはテンソル及び非テンソルを含む。
【０１３０】
例えば、正規化された形式の静的関数の入力は入力情報リストであってもく、さらに、入力情報リストは入力テンソルリスト及び属性リストを含んでもよく、属性リストは他のデータタイプの入力変数を記憶するために用いられる。一例において、入力変数のデータタイプはテンソルタイプ(ｔｅｎｓｏｒ)、整数タイプ(ｉｎｔ)、浮動小数点タイプ(ｆｌｏａｔ)を含み、ｔｅｎｓｏｒ１、ｔｅｎｓｏｒ２、ｉｎｔ、ｆｌｏａｔと表す。静的関数の入力情報リストは、テンソルリスト(ｔｅｎｓｏｒ１、ｔｅｎｓｏｒ２)及び属性リスト(ｉｎｔ、ｆｌｏａｔ)を含んでもよい。
【０１３１】
例えば、入力情報リストは、Ｃ＋＋のテンプレートパラメータ導出メカニズムを用いて取得することができる。
【０１３２】
以下、図４を参照して、本開示の一実施例による深層学習フレームワークの演算子処理方法を詳細に説明する。
【０１３３】
図４は、本開示の一実施例による深層学習フレームワークの演算子処理方法のフローチャートである。
【０１３４】
図４に示すように、方法４４１は、演算子カーネル関数入力情報と関数ポインタとの正規

化を実現することができる。以下、操作Ｓ４４１１～操作Ｓ４４１５を参照して、詳細に説明する。

【０１３５】

操作Ｓ４４１１において、登録対象演算子を取得する。

【０１３６】

例えば、登録対象演算子は、登録対象演算子記述情報及び登録対象演算子カーネル関数を含む。登録対象演算子記述情報は、登録対象演算子入力情報及び登録対象演算子出力情報を含む。登録対象演算子カーネル関数は、該演算子の主な計算ロジックを記述するために用いられる。

【０１３７】

操作Ｓ４４１２において、登録対象演算子の完全なテンプレートパラメータに対応する構造体に基づいて、少なくとも１つのサブ構造体を特定する。

【０１３８】

例えば、登録対象演算子は、複数の完全なテンプレートパラメータに基づいて作成されてもよい。各完全なテンプレートパラメータは、１つの構造体に対応してもよい。構造体は、例えば予め設定されてもよい。該構造体は、関数戻り値、可変関数パラメータリスト及び１つの関数ポインタをテンプレートパラメータとしてもよく、ここで、関数ポインタの戻り値タイプとパラメータリスト及びテンプレートパラメータにおけるタイプとが一致している。該構造体は１つの計算用の静的関数を含む。該静的関数は正規化された形式を有する。

【０１３９】

また例えば、該構造体は例えば少なくとも１つの特化サブ構造体を含んでもよい。特化サブ構造体はテンプレートパラメータ導出を行うために用いられる。

【０１４０】

操作Ｓ４４１３において、１つの特化サブ構造体と登録対象演算子の入力情報とをマッチングして、構造体の静的関数の入力サブ情報を得る。

【０１４１】

例えば、特定のデータタイプの構造体にマッチングするように、各特化サブ構造体の特化実装を予め設定することができる。各サブ構造体の特化実装は、いずれも該構造体にマッチングするデータタイプを最初のテンプレートパラメータとし、その後、残りのテンプレートパラメータリストとする。

【０１４２】

また例えば、少なくとも１つの特化サブ構造体は、終了マーカを有する１つの特化サブ構造体を含む。該終了マーカを有する特化サブ構造体は、終了マーカをテンプレートパラメータとすることができる。

【０１４３】

また例えば、特化サブ構造体はサブ静的関数を含んでもよい。サブ静的関数はパラメータインデックスをテンプレートパラメータとし、現在何番目のパラメータにマッチングしたかをマークすることに用いられる。

【０１４４】

操作Ｓ４４１４において、終了マーカにマッチングしたか否かを判定する。

【０１４５】

終了マーカにマッチングした場合、操作Ｓ４４１５を実行し、フローを終了する。

【０１４６】

終了マーカにマッチングしなかった場合、操作Ｓ４４１３に戻り、次のサブ構造体に基づいてマッチングする。

【０１４７】

例えば、コンパイル過程において、コンパイラは、演算子カーネル関数の入力情報を一つずつ解析する。現在解析されている入力情報が前述定義されたサブ構造体のある特化実装の第１個のテンプレートパラメータとマッチングした場合、該特化サブ構造体の特化実装

を呼び出し、現在パラメータに対する解析を完了する。次に、引き続き残りの入力情報を他の特化サブ構造体とマッチングして、後の入力情報を解析し、終了マーカをテンプレートパラメータとする特化サブ構造体とマッチングした場合、マッチングが完了する。

【０１４８】

さらに、幾つかの実施例において、正規化された形式を有する関数の関数形式は、以下の通りであってもよい。

【０１４９】

Ｔｅｎｓｏｒリスト演算子計算関数名(入力Ｔｅｎｓｏｒリスト){ ...}を返す。

【０１５０】

図５は、本開示の別の実施例による深層学習フレームワークの演算子処理方法のフローチャートである。

【０１５１】

操作Ｓ５１０において、処理対象演算子を取得する。

【０１５２】

例えば、処理対象演算子は、演算子カーネル関数及びテンプレートパラメータを含む。該テンプレート関数は、深層学習フレームワークに依存しておらず、処理対象演算子を配布することに用いられる。

【０１５３】

次に、方法５００は、処理対象演算子に対する入力情報を受信したことに応じて、入力情報を用いてテンプレートパラメータを解析して、深層学習フレームワークに依存する複数の完全なテンプレートパラメータを得る。以下、操作Ｓ５２１～操作Ｓ５２３を参照して詳細に説明する。

【０１５４】

操作Ｓ５２１において、マクロを用いて可変パラメータリストを作成する。

【０１５５】

操作Ｓ５２２において、可変パラメータリストを用いてテンプレートパラメータを反復的に解析して、複数の内部テンプレートパラメータを得る。

【０１５６】

操作Ｓ５２３において、複数の内部テンプレートパラメータをスプライシングして、複数の完全なテンプレートパラメータを得る。

【０１５７】

理解できることとして、操作Ｓ５２２及び操作Ｓ５２３の詳細な説明について、上述した操作Ｓ２２１１及び操作Ｓ２２１２を参照することができ、本開示はここで説明を繰り返さない。

【０１５８】

次に、方法５００は、複数の完全なテンプレートパラメータに基づいて、演算子カーネル関数を処理して、深層学習フレームワーク用の利用可能な演算子を得ることができる。以下、操作Ｓ５３１、Ｓ５３２及び操作Ｓ５３３を参照して詳細に説明する。

【０１５９】

操作Ｓ５３１において、複数の完全なテンプレートパラメータを明示的にインスタンス化する。

【０１６０】

本開示の実施例において、複数の完全なテンプレートパラメータの各々は、テンプレート関数記述情報及びテンプレート関数情報を含む。

【０１６１】

例えば、テンプレート関数記述情報をヘッダファイルとしてインスタンス化する。テンプレート関数記述情報は、テンプレート関数宣言であってもよい。ヘッダファイルは、例えば「.ｈ」ファイルであってもよい。

【０１６２】

例えば、テンプレート関数情報をソースコードファイルとしてインスタンス化する。テン

プレート関数情報は、テンプレート関数実装であってもよい。ソースコードファイルは、例えば「.ｃｃ」ファイルであってもよい。

【０１６３】

Ｃ＋＋プログラミング言語において、テンプレート関数を適用するカーネル形式には一定の制限がある。コンパイル時に、テンプレート関数は、実際のテンプレートパラメータタイプに基づいてインスタンス化し、テンプレート関数の実装も対応する呼び出し位置に展開する必要がある。テンプレート関数記述情報をヘッダファイルにインスタンス化し、ソースコードファイルをソースコードファイルにインスタンス化することで、コンパイル時に大量のテンプレート関数コードを導入して展開することによりコンパイル効率に影響することを回避することができる。

【０１６４】

操作Ｓ５３２において、完全なテンプレートパラメータの入力情報を対応するパラメータ情報に変換する。

【０１６５】

理解できることとして、操作Ｓ５３２の詳細な説明について、上述した方法３３０を参照することができ、本開示はここで説明を繰り返さない。

【０１６６】

操作Ｓ５３３において、登録対象演算子カーネル関数の入力情報及び関数ポインタの正規化を実現する。

【０１６７】

理解できることとして、操作Ｓ５３３の詳細な説明について、上述した方法４４１を参照することができ、本開示はここで説明を繰り返さない。

【０１６８】

操作Ｓ５４０において、演算子を深層学習フレームワーク内部のグローバル演算子テーブルに登録する。

【０１６９】

例えば、操作Ｓ５３２又は操作Ｓ５３３を実行した後、演算子をグローバル演算子テーブルに登録する情報を取得することができる。これらの情報は、例えば演算子カーネル関数の対応するパラメータ情報及び／又は正規化された関数ポインタを含んでもよい。

【０１７０】

操作Ｓ５５０において、グローバル演算子テーブルに基づいて、利用可能な演算子を呼び出す。

【０１７１】

幾つかの実施例において、上述したソースコードファイルにおいて、テンプレート関数を明示的にインスタンス化することができる。

【０１７２】

例えば、ヘッダファイルにおいて、テンプレート関数記述情報は、例えば以下の通りであってもよい。

【０１７３】

【数７】

```
template <typename T, typename Context>
void ScaleKernel(const Context& dev_ctx,
                 const DenseTensor& x,
                 const Scalar& scale,
                 float bias,
                 bool bias_after_scale,
                 DenseTensor* out)。
```

【０１７４】
また例えば、ソースコードファイルにおいて、明示的にインスタンス化された関数ＳｃａｌｅＫｅｒｎｅｌ〈ｆｌｏａｔ、 ＣＰＵＣｏｎｔｅｘｔ〉は、例えば以下の通りであってもよい。
【０１７５】
【数８】

```
template ScaleKernel<float, CPUContext>(const Context& CPUContext,
                 const DenseTensor& x,
                 const Scalar& scale,
                 float bias,
                 bool bias_after_scale,
                 DenseTensor* out)。
```

【０１７６】
データタイプが８種類であり、且つ機器種別が２種である場合、明示的にインスタンス化された関数に対応するコードは、１６組を含んでもよい。
【０１７７】
他の実施例において、複数の完全なテンプレートパラメータに基づいて、以下のような複数のインスタンス化された宣言文を生成することができる。
【０１７８】
【数９】

```
Template decltype(                                    ¥
    カーネル関数名<データタイプ,  機器コンテキストタイプ>)。
```

【０１７９】
ここで、ｄｅｃｌｔｙｐｅは、関数タイプを自動的に解析する方法である。このように、コンパイル効率を向上するとともに、書き方の間接性も向上し、この明示的なインスタンス化プロセスを自動的に完了する。
【０１８０】
幾つかの実施例において、グローバル演算子テーブルは、演算子カーネル工場クラスと、演算子カーネル名称クラス及びカーネルキー値クラスと、演算子カーネルクラスとを含む。例えば、演算子カーネルクラスは、演算子カーネル関数のポインタ、演算子カーネル関

数入力パラメータの記述情報、及び演算子カーネル関数出力パラメータの記述情報の少なくとも１つを含む。

【０１８１】

以下、図６を参照してグローバル演算子テーブルを詳細に説明する。

【０１８２】

図６は、本開示の一実施例によるグローバル演算子テーブルの例示的模式図である。

【０１８３】

図６に示すように、グローバル演算子テーブル６００のデータ構造は、第１級の演算子カーネル工場クラス６１０、第２級の演算子カーネル名称クラス６２０、カーネルキー値クラス６３０、及び演算子カーネルクラス６４０を含んでもよい。

【０１８４】

図６の例示において、演算子カーネルクラス６４０は、第３級の演算子カーネル関数のポインタ６４１及び演算子カーネル関数パラメータの記述情報６４２を含んでもよい。演算子カーネル関数パラメータの記述情報６４２は、例えば演算子カーネル関数入力パラメータの記述情報及び演算子カーネル関数出力関数の記述情報を含んでもよい。

【０１８５】

幾つかの実施例において、複数の完全なテンプレートパラメータに基づいて、演算子カーネル関数を処理して、深層学習フレームワーク用の利用可能な演算子を得ることは、複数の完全なテンプレートパラメータを明示的にインスタンス化することで、演算子カーネル関数を処理して、深層学習フレームワーク用の利用可能な演算子を得ることを含む。

【０１８６】

幾つかの実施例において、複数の完全なテンプレートパラメータの各々は、テンプレート関数記述情報及びテンプレート関数情報を含み、複数の完全なテンプレートパラメータを明示的にインスタンス化することで、演算子カーネル関数を処理して、深層学習フレームワーク用の利用可能な演算子を得ることは、テンプレート関数記述情報をヘッダファイルとしてインスタンス化することと、テンプレート関数情報をソースコードファイルとしてインスタンス化することとを含む。

【０１８７】

図７は、本開示の一実施例による深層学習フレームワークの演算子処理装置のブロック図である。

【０１８８】

図７に示すように、該装置７００は、取得モジュール７１０、解析モジュール７２０、及び処理モジュール７３０を含んでもよい。

【０１８９】

取得モジュール７１０は、処理対象演算子を取得するために用いられる。例えば、処理対象演算子は、演算子カーネル関数、及び深層学習フレームワークに依存しないテンプレートパラメータを含む。

【０１９０】

解析モジュール７２０は、処理対象演算子に対する入力情報を受信したことに応じて、入力情報を用いてテンプレートパラメータを解析して、深層学習フレームワークに依存する複数の完全なテンプレートパラメータを得るために用いられる。

【０１９１】

処理モジュール７３０は、複数の完全なテンプレートパラメータに基づいて、演算子カーネル関数を処理して、深層学習フレームワーク用の利用可能な演算子を得るために用いられる。

【０１９２】

幾つかの実施例において、解析モジュールは、マクロを用いて可変パラメータリストを作成するための作成サブモジュールと、可変パラメータリストに基づいてテンプレートパラメータを反復的に解析するための解析サブモジュールとを含み、ここで、入力情報をマクロのマクロパラメータとする。

【０１９３】
幾つかの実施例において、可変パラメータリストは、処理対象演算子の名称と、演算子カーネル関数の名称と、演算子カーネル関数のデータレイアウトと、深層学習フレームワークを実装するための機器タイプと、深層学習フレームワークを実装するためのデータタイプとを含む。
【０１９４】
幾つかの実施例において、解析サブモジュールは、マクロに含まれるパラメータ数を特定するための第１の特定ユニットと、パラメータ数に応じて、反復終了位置を特定するための第２の特定ユニットと、可変パラメータリストを用いて、反復終了位置に応じて入力情報を反復に処理するための処理ユニットとを含む。
【０１９５】
幾つかの実施例において、解析サブモジュールは、可変パラメータリストを用いてテンプレートパラメータを反復に解析して、複数の内部テンプレートパラメータを得るための解析ユニットと、複数の内部テンプレートパラメータをスプライシングして、複数の完全なテンプレートパラメータを得るためのスプライシングユニットとをさらに含む。
【０１９６】
幾つかの実施例において、複数の内部テンプレートパラメータは複数の文字列を含み、スプライシングユニットは、入力情報に基づいて、複数の文字列をスプライシングして、処理対象演算子に対する複数の完全なテンプレートパラメータを得るためのスプライシングサブユニットを含む。
【０１９７】
幾つかの実施例において、処理モジュールは、複数の完全なテンプレートパラメータを明示的にインスタンス化することで、演算子カーネル関数を処理して、深層学習フレームワーク用の利用可能な演算子を得るための明示的なインスタンス化サブモジュールを含む。
【０１９８】
幾つかの実施例において、複数の完全なテンプレートパラメータの各々は、テンプレート関数記述情報及びテンプレート関数情報を含み、明示的なインスタンス化サブモジュールは、テンプレート関数記述情報をヘッダファイルとしてインスタンス化するための第１のインスタンス化ユニットと、テンプレート関数情報をソースコードファイルとしてインスタンス化するための第２のインスタンス化ユニットとを含む。
【０１９９】
幾つかの実施例において、処理モジュールは、複数の完全なテンプレートパラメータのうちの各完全なテンプレートパラメータに対して、該完全なテンプレートパラメータの関数パラメータタイプを特定するための第１の特定サブモジュールと、関数パラメータタイプに基づいて、完全なテンプレートパラメータの入力情報を対応するパラメータ情報に変換するための変換サブモジュールと、対応するパラメータ情報を記録するための記録サブモジュールとを含む。
【０２００】
幾つかの実施例において、装置７００は、複数の完全なテンプレートパラメータに基づいて、深層学習フレームワークに基づく登録対象演算子を作成するための作成モジュールと、登録対象演算子を深層学習フレームワーク内部のグローバル演算子テーブルに登録するための登録モジュールとをさらに含む。
【０２０１】
幾つかの実施例において、登録対象演算子は、登録対象演算子記述情報及び登録対象演算子カーネル関数を含み、登録対象演算子カーネル関数に正規化された入力情報及び正規化された関数ポインタが含まれ、作成モジュールは、複数の完全なテンプレートパラメータのうちの各完全なテンプレートパラメータに対して、完全なテンプレートパラメータに対応する、正規化された形式を有する静的関数を含む構造体を特定するための第２の特定サブモジュールと、静的関数の入力情報を正規化された入力情報とするための第１の正規化サブモジュールと、静的関数の関数ポインタを正規化された関数ポインタとするための第

２の正規化モジュールとを含む。
【０２０２】
幾つかの実施例において、静的関数の入力情報は入力情報リストを構成し、入力情報リストは入力テンソルリストを含み、構造体は少なくとも１つの特化サブ構造体を含み、少なくとも１つの特化サブ構造体の各々は対応するデータタイプを含み、データタイプはテンソル及び非テンソルを含む。
【０２０３】
幾つかの実施例において、グローバル演算子テーブルは、演算子カーネル工場クラスと、演算子カーネル名称クラス及びカーネルキー値クラスと、演算子カーネルクラスとを含む。
【０２０４】
幾つかの実施例において、演算子カーネルクラスは、演算子カーネル関数のポインタと、演算子カーネル関数入力パラメータの記述情報と、演算子カーネル関数出力パラメータの記述情報との少なくとも１つを含む。
【０２０５】
幾つかの実施例において、深層学習フレームワークに依存しないテンプレートパラメータは、処理対象演算子を配布するために用いられ、深層学習フレームワークに依存しないテンプレートパラメータは、深層学習フレームワークの実装機器および深層学習フレームワークのデータタイプのいずれにも依存しないテンプレートパラメータを含む。
【０２０６】
幾つかの実施例において、装置７００は、利用可能な演算子に対する呼び出しコマンドに応答して、利用可能な演算子を呼び出すための呼び出しモジュールと、呼び出された利用可能な演算子をコンパイルすることで、利用可能な演算子に対応する関数機能を実行するための実行モジュールとをさらに含む。
【０２０７】
本開示の技術案において、係れたユーザ個人情報の収集、記憶、使用、加工、伝送、提供、開示及び応用等の処理は、関連法律や法規の規定に合致しており、必要なセキュリティ対策を取っており、公序良俗に反していない。本開示の技術案において、ユーザの個人情報を取得するか又は収集する前に、いずれもユーザの許可又は同意を得た。
【０２０８】
本開示の実施例によれば、本開示は、電子機器、読取可能な記憶媒体及びコンピュータプログラムをさらに提供する。
【０２０９】
図８は、本開示の実施例を実施するための例示的電子機器８００の模式的ブロック図を示している。電子機器は、様々な形式のデジタルコンピュータを示すことを目的とし、例えば、ラップトップコンピュータ、デスクトップコンピュータ、ワークステーション、パーソナルデジタルアシスタント、サーバ、ブレードサーバ、大型コンピュータ及び他の適切なコンピュータである。電子機器は、さらに様々な形式の移動装置を示してもよく、例えば、パーソナルデジタルアシスタント、携帯電話、スマートフォン、ウェアラブル機器及び他の類似の演算装置である。本明細書に示された部材、それらの接続及び関係、並びにそれらの機能は、例示に過ぎず、本明細書に記載された及び／又は要求された本開示の実現を限定しない。
【０２１０】
図８に示すように、電子機器８００は、計算手段８０１を含み、計算手段８０１は、リードオンリーメモリ（ＲＯＭ）８０２に記憶されたコンピュータプログラム又は記憶手段８０８からランダムアクセスメモリ（ＲＡＭ）８０３にロードされたコンピュータプログラムに基づいて、様々な適切な動作及び処理を実行してもよい。ＲＡＭ８０３には、さらに電子機器８００の操作に必要な様々なプログラム及びデータを記憶してもよい。計算手段８０１、ＲＯＭ８０２、及びＲＡＭ８０３は、バス８０４を介して相互に接続される。入出力（Ｉ／Ｏ）インターフェース８０５も、バス８０４に接続される。
【０２１１】

電子機器８００における複数の部品は、Ｉ／Ｏインターフェース８０５に接続され、例えばキーボード、マウス等の入力手段８０６と、例えば様々な種類のディスプレイ、スピーカ等の出力手段８０７と、例えば磁気ディスク、光ディスク等の記憶手段８０８と、例えばネットワークカード、モデム、無線通信トランシーバ等の通信手段８０９とを含む。通信手段８０９は、電子機器８００がインターネット等のコンピュータネットワーク及び／又は各種の電気ネットワークを介して他の機器と情報・データをやり取りすることを可能にする。

【０２１２】

計算手段８０１は、処理及び演算能力を有する各種の汎用及び／又は専用の処理モジュールであってもよい。計算手段８０１の幾つかの例として、中央処理ユニット（ＣＰＵ）、ＧＰＵ（Ｇｒａｐｈｉｃｓ　Ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔ）、各種専用の人工知能（ＡＩ）演算チップ、各種機械学習モデルアルゴリズムをランニングする演算ユニット、ＤＳＰ（Ｄｉｇｉｔａｌ　Ｓｉｇｎａｌ　Ｐｒｏｃｅｓｓｏｒ）、並びに任意の適切なプロセッサ、コントローラ、マイクロコントローラ等が挙げられるが、これらに限定されない。計算手段８０１は、前文で記載された各方法及び処理、例えば深層学習フレームワークの演算子処理方法を実行する。例えば、幾つかの実施例において、深層学習フレームワークの演算子処理方法は、例えば記憶手段８０８のような機械可読媒体に有形的に含まれるコンピュータソフトウェアプログラムとして実現されてもよい。いくつかの実施例において、コンピュータプログラムの一部又は全部は、ＲＯＭ　８０２及び／又は通信手段８０９を介して電子機器８００にロード及び／又はインストールされてもよい。コンピュータプログラムがＲＡＭ８０３にロードされて計算手段８０１により実行される場合、前文に記載の深層学習フレームワークの演算子処理方法の１つ又は複数のステップを実行してもよい。代替的に、他の実施例において、計算手段８０１は、他の任意の適切な方式（例えば、ファームウェアを介する）により深層学習フレームワークの演算子処理方法を実行するように構成されてもよい。

【０２１３】

本明細書で以上に説明されたシステム及び技術の様々な実施形態は、デジタル電子回路システム、集積回路システム、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、特定用途向け集積回路（ＡＳＩＣ）、特定用途向け標準製品（ＡＳＳＰ）、システムオンチップ（ＳＯＣ）、コンプレックスプログラマブルロジックデバイス（ＣＰＬＤ）、コンピュータハードウェア、ファームウェア、ソフトウェア、及び／又はそれらの組み合わせにおいて実現されてもよい。これらの様々な実施形態は、１つ又は複数のコンピュータプログラムにおいて実施され、該１つ又は複数のコンピュータプログラムは、少なくとも１つのプログラマブルプロセッサを含むプログラマブルシステムで実行され及び／又は解釈されることが可能であり、該プログラマブルプロセッサは、専用又は汎用のプログラマブルプロセッサであってもよく、記憶システム、少なくとも１つの入力装置、及び少なくとも１つの出力装置からデータ及び命令を受信し、かつデータ及び命令を該記憶システム、該少なくとも１つの入力装置、及び該少なくとも１つの出力装置に伝送することができることを含んでもよい。

【０２１４】

本開示の方法を実施するためのプログラムコードは、１つ又は複数のプログラミング言語の任意の組み合わせで作成されてもよい。これらのプログラムコードは、汎用コンピュータ、専用コンピュータ又は他のプログラマブルデータ処理装置のプロセッサ又はコントローラに提供されてもよく、それによって、プログラムコードがプロセッサ又はコントローラにより実行される時に、フローチャート及び／又はブロック図に規定された機能・操作が実施される。プログラムコードは、機器に完全に実行されてもよく、部分的に機器で実行されてもよく、独立したソフトウェアパッケージとして部分的に機器で実行され、かつ部分的に遠隔機器で実行されるか又は完全に遠隔機器又はサーバで実行されてもよい。

【０２１５】

本開示のコンテキストにおいて、機械可読媒体は、有形の媒体であってもよく、命令実行

システム、装置又は電子機器に使用され、又は命令実行システム、装置又は電子機器と組み合わせて使用されるプログラムを含んで又は記憶してもよい。機械可読媒体は、機械可読信号媒体又は機械可読記憶媒体であってもよい。機械可読媒体は、電子の、磁気的、光学的、電磁的、赤外線の、又は半導体システム、装置又は電子機器、又は上記内容の任意の適切な組み合わせを含んでもよいが、それらに限定されない。機械可読記憶媒体のより具体的な例としては、１つ以上の線による電気的接続、携帯式コンピュータディスク、ハードディスク、ランダムアクセスメモリ（ＲＡＭ）、読み出し専用メモリ（ＲＯＭ）、消去可能なプログラマブルリードオンリーメモリ（ＥＰＲＯＭ又はフラッシュメモリ）、光ファイバ、コンパクトディスクリードオンリーメモリ（ＣＤ-ＲＯＭ）、光学記憶装置、磁気記憶装置、又は上記内容の任意の適切な組み合わせを含む。

【０２１６】

ユーザとの対話を提供するために、コンピュータにここで説明されたシステム及び技術を実施させてもよく、該コンピュータは、ユーザに情報を表示するための表示装置（例えば、ＣＲＴ（陰極線管）又はＬＣＤ（液晶ディスプレイ）モニタ）と、キーボード及びポインティングデバイス（例えば、マウス又はトラックボール）とを備え、ユーザは、該キーボード及び該ポインティングデバイスを介して入力をコンピュータに提供することができる。他の種類の装置は、さらにユーザとの対話を提供してもよく、例えば、ユーザに提供されたフィードバックは、いかなる形式のセンシングフィードバック（例えば、視覚フィードバック、聴覚フィードバック、又は触覚フィードバック）であってもよく、かついかなる形式（音声入力、語音入力又は、触覚入力を含む）でユーザからの入力を受信してもよい。

【０２１７】

ここで説明されたシステム及び技術は、バックグラウンド部品を含むコンピューティングシステム（例えば、データサーバとする）、又はミドルウェア部品を含むコンピューティングシステム（例えば、アプリケーションサーバ）、又はフロントエンド部品を含むコンピューティングシステム（例えば、グラフィカルユーザインタフェース又はウェブブラウザを有するユーザコンピュータ、ユーザが該グラフィカルユーザインタフェース又は該ネットワークブラウザを介してここで説明されたシステム及び技術の実施形態と対話することができる）、又はこのようなバックグラウンド部品、ミドルウェア部品、又はフロントエンド部品のいずれかの組み合わせを含むコンピューティングシステムに実施されることが可能である。任意の形式又は媒体のデジタルデータ通信（例えば、通信ネットワーク）によりシステムの部品を互いに接続することができる。通信ネットワークの例としては、ローカルエリアネットワーク（ＬＡＮ）、ワイドエリアネットワーク（ＷＡＮ）及びインターネットを例示的に含む。

【０２１８】

コンピュータシステムは、クライアント及びサーバを含んでもよい。クライアントとサーバ同士は、一般的に離れており、通常、通信ネットワークを介して対話する。クライアントとサーバとの関係は、該当するコンピュータ上でランニングし、クライアント－サーバの関係を有するコンピュータプログラムによって生成される。サーバは、クラウドサーバであってもよく、分散型システムのサーバであってもよく、又はブロックチェーンを組合せたサーバであってもよい。

【０２１９】

理解されるべきこととして、以上に示された様々な形式のフローを使用してもよく、操作を改めてソーティングしたり、追加したり又は削除してもよい。例えば、本開示に記載の各操作は、並列に実行されたり、順次に実行されたり、又は異なる順序で実行されてもよく、本開示に開示された技術案の所望の結果を実現することができれば、本明細書はここで限定されない。

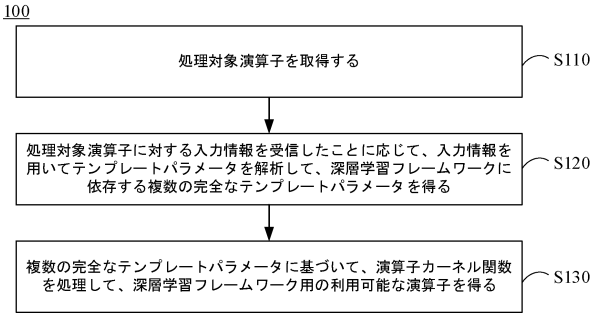
【０２２０】

上記具体的な実施形態は、本開示の保護範囲を限定するものではない。当業者であれば、設計要件及び他の要因に応じて、様々な修正、組み合わせ、サブコンビネーション及び代

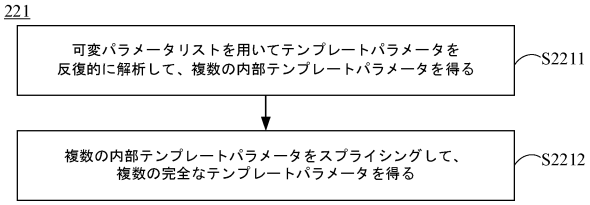
替を行うことが可能であると理解すべきである。本開示の精神と原則内で行われる任意の修正、均等置換及び改良などは、いずれも本開示の保護範囲内に含まれるべきである。

【図面】

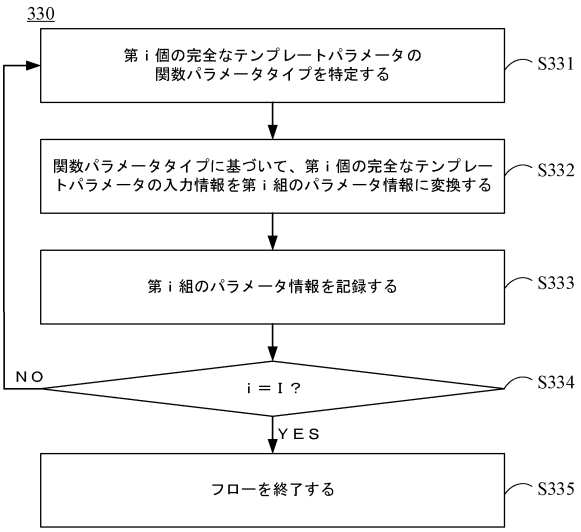
【図１】



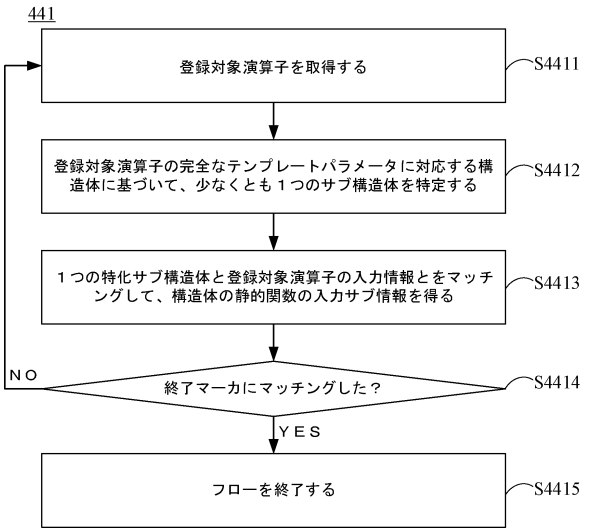

【図２】



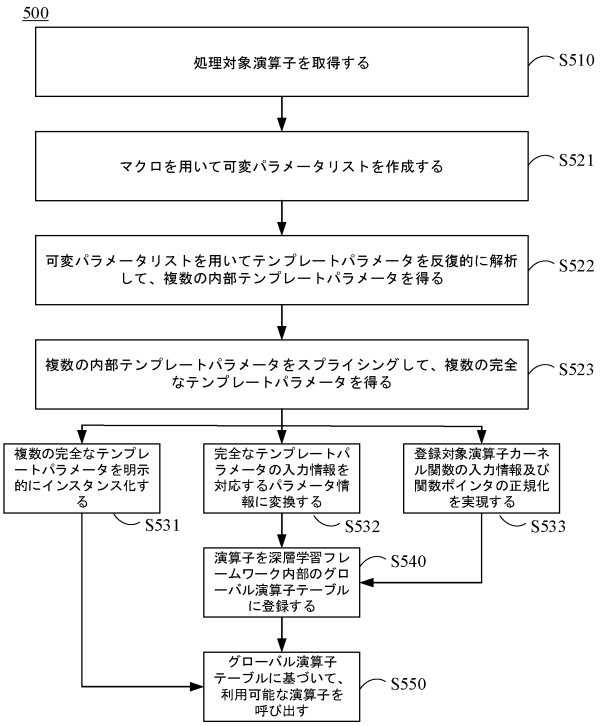

【図３】



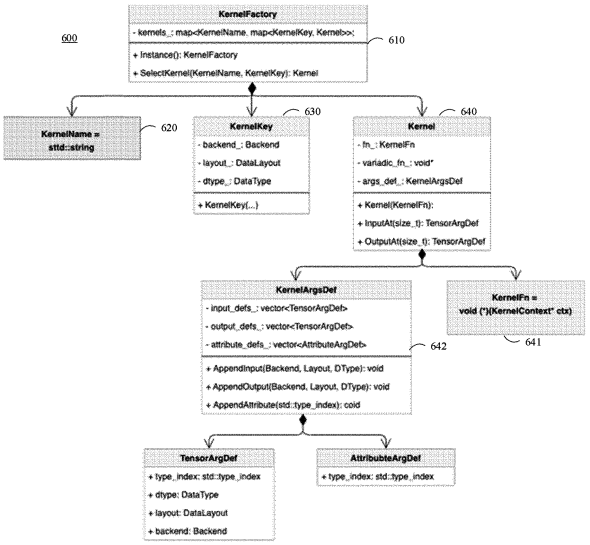

【図４】



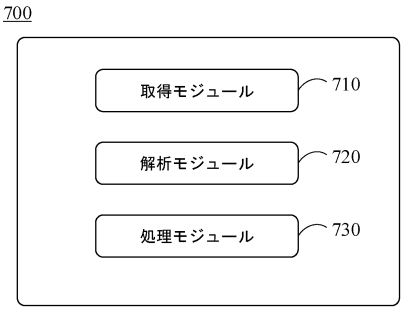
【図５】


500
処理対象演算子を取得する
S510
マクロを用いて可変パラメータリストを作成する
S521
可変パラメータリストを用いてテンプレートパラメータを反復的に解析して、複数の内部テンプレートパラメータを得る
S522
複数の内部テンプレートパラメータをスプライシングして、複数の完全なテンプレートパラメータを得る
S523
複数の完全なテンプレートパラメータを明示的にインスタンス化する
S531
完全なテンプレートパラメータの入力情報を対応するパラメータ情報に変換する
S532
登録対象演算子カーネル関数の入力情報及び関数ポインタの正規化を実現する
S533
演算子を深層学習フレームワーク内部のグローバル演算子テーブルに登録する
S540
グローバル演算子テーブルに基づいて、利用可能な演算子を呼び出す
S550

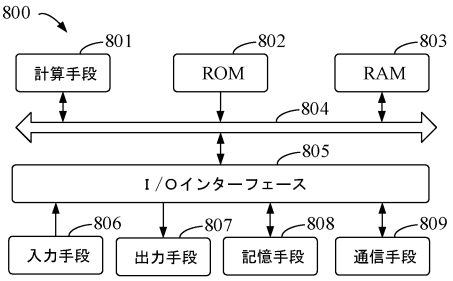
【図６】


600
KernelFactory
- kernels_: map<KernelName, map<KernelKey, KernelFn>>;
+ Instance(): KernelFactory
+ SelectKernel(KernelName, KernelKey): Kernel
610
KernelName = std::string
620
KernelKey
- backend_: Backend
- layout_: DataLayout
- dtype_: DataType
+ KernelKey(...)
630
Kernel
- fn_: KernelFn
- variadic_fn_: void*
- args_def_: KernelArgsDef
+ Kernel(KernelFn):
+ InputAt(size_t): TensorArgDef
+ OutputAt(size_t): TensorArgDef
640
KernelArgsDef
- input_defs_: vector<TensorArgDef>
- output_defs_: vector<TensorArgDef>
- attribute_defs_: vector<AttributeArgDef>
+ AppendInput(Backend, Layout, DType): void
+ AppendOutput(Backend, Layout, DType): void
+ AppendAttribute(std::type_index): void
642
KernelFn = void (*)(KernelContext* ctx)
641
TensorArgDef
+ type_index: std::type_index
+ dtype: DataType
+ layout: DataLayout
+ backend: Backend
AttributeArgDef
+ type_index: std::type_index

【図７】


700
取得モジュール
710
解析モジュール
720
処理モジュール
730

【図８】


800
801
計算手段
802
ROM
803
RAM
804
805
I/Oインターフェース
806
入力手段
807
出力手段
808
記憶手段
809
通信手段

フロントページの続き

８５， Ｃｈｉｎａ
(74)代理人 110000914
弁理士法人ＷｉｓｅＰｌｕｓ
(72)発明者 チェン， ウェイハン
中華人民共和国 ペキン １０００８５， ハイディアン ディストリクト， シャンディ テンス ストリート， １０番， バイドゥ キャンパス ２階
(72)発明者 ワン， ハイフォン
中華人民共和国 ペキン １０００８５， ハイディアン ディストリクト， シャンディ テンス ストリート， １０番， バイドゥ キャンパス ２階
(72)発明者 ジャン， ユンフェイ
中華人民共和国 ペキン １０００８５， ハイディアン ディストリクト， シャンディ テンス ストリート， １０番， バイドゥ キャンパス ２階
(72)発明者 ユェン， リーシオン
中華人民共和国 ペキン １０００８５， ハイディアン ディストリクト， シャンディ テンス ストリート， １０番， バイドゥ キャンパス ２階
(72)発明者 チェン， ティエンユー
中華人民共和国 ペキン １０００８５， ハイディアン ディストリクト， シャンディ テンス ストリート， １０番， バイドゥ キャンパス ２階
(72)発明者 リウ， ホンユー
中華人民共和国 ペキン １０００８５， ハイディアン ディストリクト， シャンディ テンス ストリート， １０番， バイドゥ キャンパス ２階
(72)発明者 フー， シャオグァン
中華人民共和国 ペキン １０００８５， ハイディアン ディストリクト， シャンディ テンス ストリート， １０番， バイドゥ キャンパス ２階
(72)発明者 ユー， ディエンハイ
中華人民共和国 ペキン １０００８５， ハイディアン ディストリクト， シャンディ テンス ストリート， １０番， バイドゥ キャンパス ２階
(72)発明者 マー， イェンジュン
中華人民共和国 ペキン １０００８５， ハイディアン ディストリクト， シャンディ テンス ストリート， １０番， バイドゥ キャンパス ２階

審査官 宮司 卓佳
(56)参考文献 中国特許出願公開第１１３３４２３４６（ＣＮ，Ａ）
米国特許出願公開第２０１９／０３９２２９６（ＵＳ，Ａ１）
国際公開第２０２０／２６３４２１（ＷＯ，Ａ１）
中国特許出願公開第１１３７０５７９８（ＣＮ，Ａ）
中国特許出願公開第１１１３１０９３４（ＣＮ，Ａ）
米国特許第１０９５６１３２（ＵＳ，Ｂ１）
中国特許出願公開第１１３０３１９６６（ＣＮ，Ａ）
(58)調査した分野 (Int.Cl.，ＤＢ名)
Ｇ０６Ｎ ３／１０
Ｇ０６Ｆ ８／４０
Ｇ０６Ｆ ８／３０