Vol.2017-HPC-162 No.24
2017/12/19

疎行列形状のカラー画像を入力としたディープラーニングによる数値計算ライブラリの自動チューニング方式

山田　賢也†, 片桐　孝洋††, 永井　亨††, 荻野 正雄††

疎行列を対象とする数値計算ライブラリでは，実装選択に関するチューニングパラメタが多く存在するが，その設定の違いによる計算性能の差が非常に大きい．また，最適実装は計算を行う疎行列に依存する．しかし一般的に，各疎行列に対して実行して性能を調べる以外の方法で最適実装を見つけることは容易ではない．そこで本研究では，あらかじめ実行を行い最適実装が分かっている疎行列を訓練データとして利用し，ディープラーニングによる機械学習を行う最適実装の予測手法の提案と評価を行う．この際に予測対象とする実装には，数値計算ライブラリ特有の処理である，反復解法の前処理を含む．このことより，数値計算ライブラリの使用者が疎行列ベクトル積（SpMV）や反復解法の専門家でなくても，専門家による実装選択と同等，もしくは，許容範囲内の性能で実装選択できる数値計算ライブラリの構築を目指す．

1. はじめに

疎行列は数値計算の様々な分野で取り扱われており，疎行列を対象とする数値計算ライブラリ（疎行列ライブラリ）が広く利用されている．ここで，数値計算ライブラリには実装選択に関するチューニングパラメタが多く存在するが，最適実装の選択は実行を伴う経験に基づいて行われることが多く，時間的コストが高くなりやすい問題がある．これに対し，Muralidharan ら[1]は実行を介さずに SpMV（疎行列ベクトル積）の最適実装を選択する手法を提案した．Muralidharan らの手法では，疎行列に対して行列の構造に基づく事前定義特徴を用意し，さらに機械学習のモデルとして SVM（サポートベクターマシン）を用いることで最適な SpMV 実装選択を行っている．しかし，新たな SpMV 実装方法や計算機アーキテクチャに対しては事前定義特徴を再検討する必要があり，またその検討は専門家でないと困難である．一方，Cui ら[2]は，行列の構造に基づく特徴画像を生成し，ディープラーニングによる機械学習によって最適実装選択を行う方法を提案した．ディープラーニングでは，訓練データの特徴量を学習の過程で自動的に抽出することができるため，専門家でなくとも最適実装を選択しやすくなることが期待できる．しかし，これらの研究は SpMV 実装を対象としたものであり，数値計算ライブラリ自体を対象とした研究は少ない．数値計算ライブラリを利用する場合は，反復法や前処理法など重要なパラメタが多く存在する．

そこで本研究では，専門家でなくても最適な実装方式が自動設定できる数値計算ライブラリの構築を目的に，人工知能技術による実装選択手法の開発を目指す．特に，疎行列のカラー画像に基づく特徴画像の構築手法を提案し，数値実験による提案手法の効果検証を行う．本論文の構成を以下に示す．2 章で本研究において取り扱う行列やライブラリ等についての説明を行う．3 章で実験の手順や問題設定，評価方法についての詳細な説明を行う．4 章で実験結果の提示と評価を行い，最後に 5 章で結論を述べる．

2. データ及びライブラリ

2.1 数値計算ライブラリ Xabclib

本研究では疎行列を対象とした数値計算ライブラリである Xabclib [3]を利用する．特に，連立 1 次方程式の数値解を反復法である GMRES 法で求める Xabclib_GMRES を対象とする．Xabclib_GMRES では SpMV 実装方式，前処理法，リスタート周期等のいわゆるチューニングパラメタが複数存在するが，今回は本手法の効果の検証のために，SpMV 実装方式と前処理法の 2 種類に絞って行う．

2.2 利用する疎行列データセット

本研究で利用する疎行列データは，フロリダ大学疎行列コレクション[4]中の正方行列すべてとする．ただし，実際の数値実験では，後述する条件を満たさない疎行列は使用するデータセットから外して行っている．ここで，本研究で利用する疎行列データのまとまりを，疎行列データセットと呼ぶ．

2.3 ディープラーニングフレームワーク

本研究ではディープラーニングフレームワークとして TensorFlow[5]を利用する．TensorFlow はデータフローグラフを使用した数値計算用のオープンソースソフトウェアライブラリである. 柔軟なアーキテクチャにより, 単一の API で様々なプラットホーム上の CPU や GPU で計算を行うことが可能である．もともとは機械学習やディープニューラルネットを研究する目的で開発されたものだが，そのシステムの汎用性が高く，多分野で利用されている．

主な対象プログラミング言語は Python であるが，C/C++ にも対応している．対応 OS は Ubuntu/Linux 64bit または Mac OS X である．加えて，Android，iOS，Raspberry Pi 向けのコードも公開されている．

† 名古屋大学 大学院情報学研究科情報システム学専攻
†† 名古屋大学 情報基盤センター 大規模計算支援環境研究部門

ⓒ2017 Information Processing Society of Japan

3. 実験手順

3.1 概要

本研究では，1.1 節で記述した Cui らの手法（以降，既存手法と記述する）を数値計算ライブラリに適用するとともに，一部新たな手法を加えて実用可能であるかどうかの検討を行う．この章ではその実験手順と問題設定について説明する．実験は大まかに以下の手順に従って行う．

1. 各疎行列に対して Xabclib_GMRES を実行し，最適な実装を調べる．
2. 各疎行列を各手法に従って画像に変換し，特徴画像を得る．
3. 1，2 で行った作業で得られた最適実装と特徴画像を用いて訓練データとテストデータを作成する．
4. 訓練データを用いてディープラーニングによる機械学習を行う．
5. 学習が完了した後のモデルとテストデータを使用して予測結果を得る．
6. 結果を評価する．

それぞれの詳細は，次節以降で説明を行う．

3.2 各手順の詳細

3.2.1 最適実装の取得

本節では，各疎行列に対して Xabclib_GMRES を実行する際の最適な実装の取得と，その際の実験設定について説明を行う．

疎行列データセットに対して Xabclib_GMRES を実行し，最速計算を行える実装を調べた．以下，実験設定について記述を行う．Xabclib_GMRES は式（3.1）に示す係数と定数項が実数である連立 1 次方程式の数値解を求める GMRES 法を実装したものであるが，今回の実験では式(3.1)の左辺の行列 A は計算を行う各疎行列である．右辺ベクトル b は，解となるベクトル x の要素が全て 1.0 となるように値を設定した．

$$Ax = b \qquad (3.1)$$

Xabclib_GMRES では，様々な実装選択のチューニングパラメタが存在する．今回の実験では，SpMV 実装方式と前処理選択の 2 種類のチューニングに限って実験を行うことは前述したとおりであるが，その詳細を表 1 及び 2 に示す．各表にある通り，前処理 6 種類，SpMV 実装 4 種類を予測する実装の対象とし，これらの組み合わせである 24 通りの実装に対して最適実装の予測を行う．

表 1:Xabclib_GMRES の前処理実装一覧

前処理		
前処理なし	SSOR	ILU(0)
Jacobi	ILU(0)_Diagonal	ILUT

表 2:Xabclib_GMRES の SpMV 実装一覧

SpMV 実装	
Row Decomposition Method	Branchless Segmented Scan
Normalized NZ method	Original Segmented Scan

これらの設定に加えて，今回の実験では，反復法は収束解を得ることが重要であることから，高性能な最適実装選択を目的として，実行時間制限 600 秒，リスタート回数制限 1000 回，収束条件は相対残差が1.0×10^{-8}以下の条件のもとで Xabclib_GMRES を実行した．この条件の下で収束解が一つも得られなかった疎行列は，使用データから除外した．これにより，最終的に使用する疎行列データセットは，次元数が 216 から 256,000 まで，および，非ゼロ要素の割合が約 0.0004%から 56%の範囲である 1058 個となった．

3.2.2 既存手法における特徴画像への変換

本節では，各疎行列を直接的に画像へと変換する方法について，その中でも既存手法についての説明を行う．なお，これ以降，特徴画像とはディープラーニングに使用するための疎行列画像を指す．次節で詳細は説明するが，今回の実験で用いたニューラルネットは CNN（畳み込みニューラルネット）である．畳み込みニューラルネットでは画像を入力として使用する場合，全て同じサイズの画像でなくてはならない．ここで，入力として疎行列の要素を画像のピクセルに対応させるような単純な方法で画像化して用いようとすると，疎行列のサイズによって画像のサイズが変わり，入力として使用することができなくなる．

そこで，まず考えられるのは，上記方法で作成した行列画像のリサイズを行って固定サイズの画像とする方法である．これにより，CNN の入力データとして使用する上での問題点は改善される．しかしながら，この単純な方法では，行列サイズの情報が失われてしまうため，何らかの工夫を加える必要がある．

そこで，既存手法では color_index という概念を導入した．color_index とは，行列の次元を正規化し，0 から 255 までの整数値に変換した値である．具体的には以下の式（3.2）に従った color_index を得る．ただし以下の Size は画像に変換する行列のサイズを示し，$\mathrm{Size}_{\max}$，$\mathrm{Size}_{\min}$は，それぞれ扱う全ての行列における行列のサイズの最大値と最小値を示す．

$$\mathrm{color}_{\mathrm{index}} = \left\lfloor \frac{\mathrm{Size} - \mathrm{Size}_{\min}}{\mathrm{Size}_{\max} - \mathrm{Size}_{\min}} \times 255 \right\rfloor \qquad (3.2)$$

以上をふまえて，具体的な特徴画像の生成は以下の手順で行う．

1. Size×Size の大きさのグレースケール画像を生成する．この際，画素値はすべて 0 で初期化する．
2. 変換元行列において非ゼロ要素がある場所を i 行 j 列

ⓒ2017 Information Processing Society of Japan

とすると, 画像内の座標（i, j）に color_index を画素値として挿入する.

3. 2 の手順を変換元行列に存在するすべての非ゼロ要素に対して行う.（これにより，挿入された全ての画素値は一定の値（= color_index）となる）.
4. OpenCV[6] の resize（）関数を用いて固定サイズの画像にリサイズする（今回の実験では 32×32，補完方法はバイリニア補間）.

以上の手順で得られた既存手法による特徴画像の例（行列名:xingo_afonso_itaipu）を図 1 に示す.

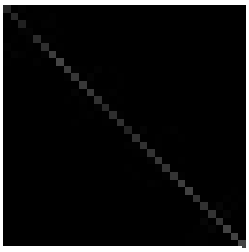


図 1:既存手法による特徴画像例

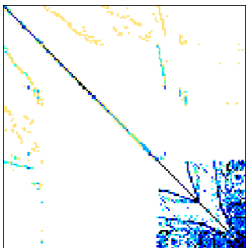


図 2:xingo_afonso_itaipu 行列画像

3.2.3 提案手法における特徴画像への変換

前節では既存手法における特徴画像の生成方法を示した. 本節では，各疎行列を直接的に画像へと変換する方法について，提案手法の説明を行う.

前述の通り既存手法では最適実装の予測を行う対象を SpMV としており，数値計算ライブラリ全体の機能についての実装予測は考えられていない. そのため SpMV 実装の予測に必要となる疎行列の非ゼロ要素の位置（構造情報）は特徴画像に残っているが，SpMV 実装の予測に必要ではない非ゼロ要素の値の情報は特徴画像に残っていない. 本実験での実装予測は前処理方式の予測も含まれており，非ゼロ要素の値も実装予測に必要な情報であるため，このままでは予測精度が低いことが考えられる. 加えて，図 1 は xingo_afonso_itaipu 行列を例として生成された特徴画像を示しているが，この行列の画像をフロリダ大学疎行列コレクションのサイト[7]から引用した図が図 2 である. この行列画像では，色のついている部分に非ゼロ要素が存在している. これと既存手法による図を比較すると，既存手法による図では，対角要素以外の部分の非ゼロ要素が消えている箇所がある. これはリサイズの際のサイズ比（今回の例では 13250:32）が大きいため，生じると考えられる.

以上の 2 点を解消するために，提案手法として以下の 2 つの手法を適用して特徴画像を生成する.

① 特徴画像生成の際に既存手法のように OpenCV の resize（）関数を使うのではなく，元行列を 32×32 等分に分割し，その中に非ゼロ要素が存在すれば特徴画像の対応する座標に画素値を与える方法を利用する. これによりリサイズ比が大きい場合であってもその要素を落とすことなく特徴画像に反映させることができる.

② 画像に残す情報の種類を増やすために既存手法のようなグレースケールの画像ではなく，RGB3 色からなるカラー画像を用いて特徴画像を生成する. これによりグレースケール画像では 1 種類しか残すことのできなかった情報を 3 種類増やすことが可能となり, 非ゼロ要素の値の情報を残すことが可能となる.

まず①について，具体的には以下の手順に従って画像を生成する.

1. 32×32 の RGB カラー画像を生成する.
2. 元行列を図 3 のようにリサイズ後のサイズである 32×32 等分に区画分けを行う.（なお図 3 は例であり 31×31 に区画分けがされているが実際は 32×32 等分に区画分けを行う.）
3. 元行列において非ゼロ要素が 1 つ以上存在する区画を（i, j）とすると，後述する画素値生成方法により，画素値R_{ij}, G, Bを生成する（図 3 内の赤円）.
4. 変換中のカラー画像内の対応する座標（i, j）に画素値を挿入する（図 4 内の赤四角）.
5. 3,4 の手順を繰り返し，元行列の非ゼロ要素が存在する全ての区画に対してこれらの処理を行う.（非ゼロ要素が存在しない区画では何も行わない（図 3 及び 4 内の緑円）.）

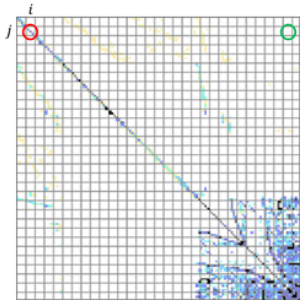


図 3: 変換元行列

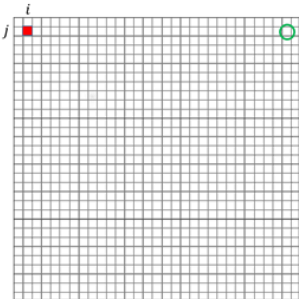


図 4: 変換中カラー画像

次に②について，カラー画像を使用して特徴画像を生成する際にそれぞれの色にどのような情報を当てはめるか，及び各色の画素値をどのように設定するか，について説明を行う.

● 青色について

青色は既存手法（グレースケール画像）の場合でも情報として残してある行列サイズの情報を挿入することとした. 手法としては前述の既存手法における color_index を用いる場合と同様に以下の式（3.3）に従って青色の画素値を設定した. ただし以下の B は青の画素値を示す.

$$B = \left\lfloor \frac{Size - Size_{min}}{Size_{max} - Size_{min}} \times 255 \right\rfloor \quad (3.3)$$

ⓒ2017 Information Processing Society of Japan

なお青色の画素値は，一つの特徴画像内で全て同じ B の値を利用する．

- 緑色について

①の手法により OpenCV を用いない固定サイズの画像生成が行えるようになったが，非ゼロ要素の密度が薄い部分も残さず画像に反映させる影響により，画像化後に全体として非ゼロ要素の密度が濃くなってしまう．そのため，緑色にはその変化を補完する目的として疎度の情報を挿入することとした．

画素値の決定に際して，まず以下の式（3.4）に従って得られた疎度（Spa）を得る．以下の nnz は画像に変換する行列の非ゼロ要素の数を示す．

$$\mathrm{Spa} = \frac{\mathrm{nnz}}{\mathrm{Size}^2} \tag{3.4}$$

式(3.4)を用いて緑の画素値を算出するのだが，Spa の値域は$0 < \mathrm{Spa} \leq 1$である点と，疎度の値が特に値が 0 に近い場合に多くの値が分布しているという点から，疎度を対数にとったうえで画素値を計算するようにした．これにより，疎度の差が色の変化に出やすくなるようになる．具体的には以下の式（3.5）に従って画素値を設定する．ただし，式(3.5)の G は緑の画素値，$\mathrm{Spa}_{\min}$は扱う全ての行列における行列の疎度の最小値を示す．

$$\mathrm{G} = \left\lfloor \frac{\log_2 Spa}{\log_2 Spa_{min}} \times 255 \right\rfloor \tag{3.5}$$

なお緑色の画素値も青色と同様に，一つの特徴画像内で全て同じ G の値を利用する．

- 赤色について

前述の通り数値計算ライブラリの最適実装予測において非ゼロ要素の値は欠かすことのできないものである．そのため赤色には非ゼロ要素の値の情報を挿入することとした．

非ゼロ要素の値は疎行列内のそれぞれの要素で異なるものであるので，各座標の非ゼロ要素の値に応じて画素値を定める必要がある．①の手法では元行列の区画分けを行っているため，それぞれの区画内における非ゼロ要素の値に応じて赤色の画素値を設定すればよい．そこで以下のような手順をとる．

まず変換する疎行列の全ての非ゼロ要素を以下の式（3.6）に従い絶対値をとる．ただし以下の$\mathrm{Val}_{\mathrm{kl}}$は元行列内の k 行 l 列における非ゼロ要素の値を示し，この値に対して絶対値をとった後の値を$\mathrm{Val_abs}_{\mathrm{kl}}$とする．

$$\mathrm{Val_abs}_{\mathrm{kl}} = |\mathrm{Val}_{\mathrm{kl}}| \tag{3.6}$$

次に前述の通りに変換元行列を区画分けし，区画毎に$\mathrm{Val_abs}_{\mathrm{kl}}$を合計し，その区画内の非ゼロ要素の数で割って平均をとることで，区画（i, j）におけるこの非ゼロ要素の値の大きさを算出することが可能である．直前の手順ですべての非ゼロ要素の値に絶対値をとったのは，この手順を行う際に正の値と負の値が区画内に混在しているとお互いに打ち消しあって値が失われてしまうため，これを防ぐためである．しかし，このまま画素値を生成すると非ゼロ要素の値の値域が広く，かつ値の分布が値の小さいものと大きいものに偏っている様な場合に色の変化が出にくい場合がある．

そこで，$\mathrm{Val_abs}_{\mathrm{kl}}$は絶対値をとっており，必ず 0 以上の値となっていることから，$\mathrm{Val_abs}_{\mathrm{kl}}$の最大値と最小値の差が 255 よりも大きい場合に，緑色における疎度の場合と同様に対数をとることで色の変化を出やすくする．この際，単純に絶対値をとるだけでは$\mathrm{Val_abs}_{\mathrm{kl}}$が 1 未満の場合で負の値が発生してしまう．これでは区画内での値の打消しが再び発生してしまう．この点を解消するために，$\mathrm{Val_abs}_{\mathrm{kl}} \leq 1$の範囲では対数を取らず，値を 0 として扱う処理も加える．

具体的には以下の式（3.7）に従い，変換後の非ゼロ要素の値$\mathrm{Log_abs}_{\mathrm{kl}}$を得る．

$$\begin{cases} \mathrm{Log_abs}_{\mathrm{kl}} = \log_2 \mathrm{Val}_{\mathrm{abs}_{\mathrm{kl}}} & (1 < \mathrm{Val_abs}_{\mathrm{kl}}) \\ \mathrm{Log_abs}_{\mathrm{kl}} = 0 & (\mathrm{otherwise}) \end{cases} \tag{3.7}$$

この処理を行ったことにより，色の変化が出やすくなったため，前述のとおり区画毎に$\mathrm{Log_abs}_{\mathrm{kl}}$を合計し，その区画内の非ゼロ要素の数で割ることで区画ごとの非ゼロ要素の値の平均値$\mathrm{Mean}_{\mathrm{ij}}$を得る．

この値を用いて以下の式（3.8）に従って赤の画素値を得る．ただし以下のR_{ij}は座標（i, j）における赤の画素値を示し，$\mathrm{Mean}_{\max}$, $\mathrm{Mean}_{\min}$は，扱う全ての行列における$\mathrm{Mean}_{\mathrm{ij}}$の最大値と最小値を示す．

$$\mathrm{R}_{\mathrm{ij}} = \left\lfloor \frac{\mathrm{Mean}_{\mathrm{kl}} - \mathrm{Mean}_{\min}}{\mathrm{Mean}_{\max} - \mathrm{Mean}_{\min}} \times 255 \right\rfloor \tag{3.8}$$

以上の①，②の手法に従って生成された特徴画像の例（行列名:xingo_afonso_itaipu）を図 5 に示す．図 1 及び 2 と比較すると構造情報が対角要素以外の部分でも残っていることに加え，非ゼロ要素の値に赤の画素値が変化していることが確認できる．

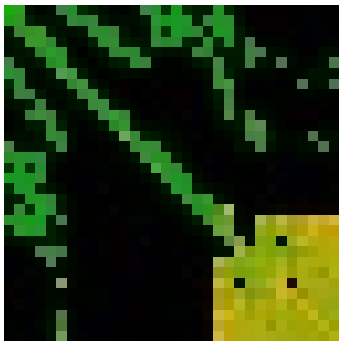


図 5: 提案手法による特徴画像例

3.3 機械学習のための訓練，テストデータの作成

前節までの手順により得られた最適実装と特徴画像の組を用いて機械学習に使用する訓練データと学習結果の評価に使用するテストデータを作成した．それぞれのデータ

ⓒ2017 Information Processing Society of Japan

の形式としては,最適実装については実装が全部で 24 通り存在することから各実装を 0 から 23 の整数値で表現し,各疎行列に対して最適なものをラベル付けした．特徴画像は 32×32 ピクセルの jpg 画像とした．

訓練データとテストデータの数の比が 4:1 となるようにデータの分割を行った．なお既存手法による特徴画像生成では OpenCV の resize（）関数を用いている。このとき元行列のサイズが大きいとこの方法によるリサイズが行えない場合があり，データ数が減少してしまう．詳細なデータ数は後述の 4 章にて結果とともに併記する．

3.4 ディープラーニングによる機械学習

本実験では，第 1 章に記述した理由から機械学習のモデルとしてディープラーニングを使用した．使用したディープラーニングのフレームワークは前述のとおり TensorFlow である．

今回ニューラルネットとして作成したのは画像認識であることから CNN（畳み込みニューラルネット）とした．その構成は，畳み込み層とプーリング層のペアが 2 組，全結合層 2 層，ソフトマックス層からなる構造である．それぞれの層の詳細な構造は, CNN の全体像を示す図 6 を参照のこと．

出力層であるソフトマックス層の出力は 24 次元のベクトルとなっているが,これらの要素にそれぞれ 24 通りの実装を一意にあてはめる．そして，各疎行列の特徴画像を入力として与えて計算を行った際に，もっとも大きい値となった要素に対応する実装を予測された実装として扱うこととする．

学習はミニバッチサイズを 50 とし,ミニバッチサイズ分の学習を 1 ステップとして 200 ステップの学習を行った．誤差関数にはクロスエントロピー，最適化には Adam 法を使用し，学習率は1.0×10^{-4}とした．全結合層の第 1 層にはドロップアウトを適用し，ドロップアウト確率は 0.5 とした． 1 ステップごとにテストデータを用いて精度を測定し，その中で最も実装予測の精度が高かった際の CNN モデルをそのデータセットにおける学習結果として扱うものとした．これは今回のデータ数を鑑みて過適合の発生が考えられるための処理である．

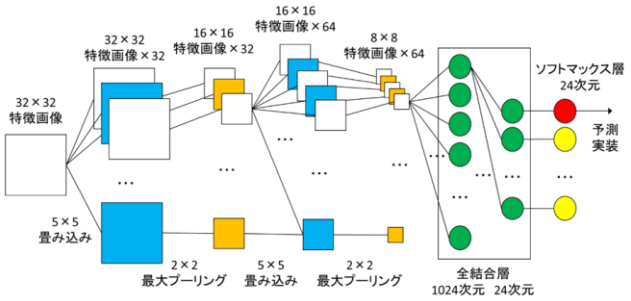


図 6:ニューラルネットワーク構造

3.5 学習結果の取得と評価

3.5.1 予備評価

前節までで説明してきた実験設定および実験手順で学習を行って得られた CNN モデルを用いてテストデータに対して最適実装予測を行い，推定された実装を記録する．そしてこの結果に対して最適な実装であるかどうかだけでなく実行時間や SpMV，前処理それぞれがどの程度最適なものを選択できているか等の観点から評価を行う．この評価に関しては 4 章にて詳細な記述を行う．なお予備評価の段階では，単純に疎行列データセットを訓練データとテストデータに 4:1 に分割しただけの場合でのみ評価を行った．

3.5.2 本評価

予備評価の段階では，疎行列データセット自身の正当性の検証が不十分であった．すなわち，予備評価の段階での疎行列データセットでは行列内の一列分だけに要素が含まれているだけの行列等が複数存在しており，検証のために使用するデータセットとしては不十分であった．

そこでそれらの疎行列の数を大幅に減らし，代わりに Xabclib_GMRES の実行制限を緩和することで,予備評価の時とほぼ同数となる 1035 個の質の高いと考えられる疎行列データセットを用意した．実行時間制限の緩和のため，実行時間制限を 600 秒から 3500 秒まで増加させることを行った．

加えて，予備評価の段階では用意した疎行列データセットに対して訓練データとテストデータへの分割を一通りしか行っていなかった．そのため，モデルの評価を行う面では不十分であった．そこで以下の評価方法を導入する．

- k-fold 交差検証

用意した疎行列データセットに対して以下の手順で精度を測定する．

1. 用意したデータセットをランダムに k 個に均等分割する．
2. テストデータとし使用していないデータセットを 1 つ選び，テストデータとし，残りのすべてのデータセットを合わせて訓練データとする．
3. 2 で設定した訓練データで学習を行い，テストデータで精度を評価する．
4. 2,3 の手順を k 回繰り返す．
5. 得られた k 個の精度の平均をとり，モデルの精度とする．

k-fold 交差検証による評価では，使用するデータセット全てを用いて精度を測定できることから，データセットの分け方によるデータの偏りを防ぐことができ，より信頼性の高い精度を求めることができる．また，モデルの汎化性能を検証したい場合に，データセットの分け方によって精

度が左右されてしまうのは好ましくないため，k-fold 交差検証を使うことで解消することが可能である．

さらに，予備評価では SpMV の予測と前処理の予測を一つの CNN モデル上で行っているが，これは必ずしも同一のモデルで行う必要はないと考えられる．むしろ予測実装種類数が 24 種類あるのに対して疎行列データセットの数は約 1000 とディープラーニングを行う上では数が不足していると考えられる．そのため，本評価ではこれらの予測を分けて行うことも行う．

以上の本評価は提案手法に対してのみ行い，結果は 4 章にて一部記載し，残りは当日の発表で説明を行う．

4. 結果と評価

4.1 実験環境

Xabclib_GMRES の実行は，名古屋大学情報基盤センター設置のスーパーコンピュータである，Fujitsu PRIMEHPC FX100 を使用した．TensorFlow は Ubuntu16.04 上にインストールされたものを使用し，CPU のみで機械学習を行った．

4.2 予備評価における実験結果

まず予測精度に関して，表 3 に既存手法による実験結果，表 4 に提案手法による実験結果示す．なお表内の最適実装予測率は SpMV 及び前処理の予測が両方とも最適なものを予測した割合を示す．最適 SpMV 実装予測率及び最適前処理予測率それぞれ SpMV 実装及び前処理選択で最適なものを予測した割合を示す．非収束実装数は推測した実装によるテストデータ行列の Xabclib_GMRES での実行が収束しないものであった行列の数を示す．

表 3:既存手法による予測精度

訓練データ数	テストデータ数	非収束実装数
755	188	57
最適実装 予測率(%)	最適 SpMV 実装予測率(%)	最適前処理 予測率(%)
42.6	71.8	58.0

表 4:提案手法による予測精度

訓練データ数	テストデータ数	非収束実装数
847	211	19
最適実装 予測率(%)	最適 SpMV 実装予測率(%)	最適前処理 予測率(%)
62.1	74.9	81.5

続いて実行時間について，表 5 に既存手法による実験結果，表 6 に提案手法による実験結果示す．なお表内の最適実行時間は，最適パラメタによるテストデータ行列の実行時間の合計，予測実行時間は予測した実装によるテストデータ行列の実行時間の合計，実行時間の差はそれらの差を示す．なお，表内の*が付与してある項目は前述の非収束実装数にカウントされている行列のデータを除いて算出している．これは実行が収束しない実装を予測しているため，実行時間が計測できないためである．

表 5:既存手法による予測実装における実行時間

最適実行 時間(s)	最適実行 時間(s)*	予測実行 時間(s)*	実行時間 の差(s)*
2925.48	1009.68	1015.64	5.966

表 6:提案手法による予測実装における実行時間

最適実行 時間(s)	最適実行 時間(s)*	予測実行 時間(s)*	実行時間 の差(s)*
4016.39	3389.18	3567.67	178.49

4.3 予備評価における評価

既存手法であるグレースケール画像の特徴画像を用いた場合と提案手法である 3 色画像の特徴画像を用いた場合で当初の推測通り前処理の予測において大きな精度改善が見受けられる．その結果，全体での最適な実装予測の正答率も高まっており，予備評価の段階では数値計算ライブラリの最適実装予測を行う場合に 3 色画像を使用することは有効であるといえる．

また最適実装ではないが，収束する実装を選べた場合では，最適実装と比較した実行時間の合計の差は既存手法においては高々1%程度，提案手法においては 6%程度であった． ただし，これは収束しない実装を選んだ場合を除いており必ずしもこの値が差の全てに相当するわけではない．しかし，既存手法では提案手法と比較してかなり多くの実行時間が非収束扱いとして除かれてしまっており，非収束のデータの数の面とも合わせて考えると，全体としての誤差としては提案手法を用いた場合の方が小さいといえる．

4.4 本評価による実験結果と評価

本評価では 5-fold 交差検証を用いて精度の測定を行った．その結果を表 7 に示す．

表 7:本評価による予測精度

最適実装 予測率(%)	最適 SpMV 実装予測率(%)	最適前処理 予測率(%)
40.0	51.4	74.8

最適実装予測率の観点で 40.0%と予備評価と比較すると 22.1 ポイント低い精度が得られた．最適 SpMV 実装予測率と前処理予測率の観点では前処理予測率においては 6.7 ポイントの精度の悪化に対して，SpMV 実装予測率では 23.5

ⓒ2017 Information Processing Society of Japan

ポイントもの大幅な精度の悪化が確認できる．これは本評価の精度が低いというよりは，予備評価の段階で疎行列データセットの質の低さにより，最適 SpMV 実装予測率が高くなってしまっていたためと推測できる．

また，疎行列データセットの数が 24 種類の実装を予測するには不足しているために精度が悪化していると考えられるため，SpMV の予測と前処理の予測の分離も行った．その場合の前処理予測率は 77.5%となり，分離しなかった場合と比較して 2.7%の改善がみられた．大幅な改善とはなっておらず，分離が有効であるかどうかについては考察が必要であるといえる．

その他の評価の詳細は当日に発表，説明を行う．

5. おわりに

本論文では，Cui らによる SpMV に対するディープラーニングを用いた最適実装選択の手法を拡張した手法を提案し，数値計算ライブラリに対して提案手法を適用し評価を行った．提案手法の特徴は，特徴画像生成の際に 3 色カラー画像を用いることで既存手法では行列サイズのみの 1 種類しか挿入できなかった情報を行列サイズ，疎度，非ゼロ要素の値の 3 種類に増やすことを可能にした点と，非ゼロ要素を可能な限り最大限特徴画像に情報として反映させた点である．

フロリダ大学疎行列コレクションを用いた実験の結果，予備評価の段階では，提案手法において最適実装を選択できなくても，収束している実装を選択できている場合であれば，収束している疎行列に対して最適パラメタによる実行時間と本提案手法による推定パラメタによる実行時間の合計の差は高々6%程度であった．この観点から，専門家でなくてもある程度の性能を得られるような実装選択が可能であると考える．したがって，数値計算ライブラリにおける最適実装予測に対してもディープラーニングによる機械学習は有効である．

しかし，予備評価の段階では，使用している疎行列データセットや評価方法が検証をする上でまだ不十分であり，より信頼性の高い検証が必要である．また，CNN の構造，及び特徴画像の生成手法に対する改善点もある．これらの要素はまだ精査できておらず，CNN の層を増やす，特徴画像サイズを大きくする，特徴画像生成の際の画素値の算出法や残す情報の変更等の改善点が考えられる．これらの変更を行った場合の評価は今後の課題である．

また，本手法用いて自動チューニング（AT）のための専用言語 ppOpen-AT [8]-[11]から利用できる AT 基盤を開発し，数値計算ライブラリ利用時に，より汎用性の高い AT 環境を構築することは今後の課題である．

謝辞

本研究の一部は，科学技術研究費補助金、基盤研究（B)、「機械学習技術の活用による職人的プログラミングの知能化」(課題番号：16H02822)，科学技術研究費補助金、挑戦的萌芽研究､「自らを進化させ未知の計算機環境に対応するソフトウェア自動チューニング機構方式の研究」(課題番号 : 15K12033)、および，日本学術振興会二国間交流事業，共同研究オープンパートナーシップ（日本-台湾）「国際交流による自動チューニングのための性能モデルの深化」による．

また，本研究に関して貴重な助言を賜りました，東北大学 滝沢寛之 教授に深く感謝致します．加えて Xabclib の利用に関し，日立製作所の櫻井隆雄氏に深く感謝致します．

参考文献

1) Saurav Muralidharan, Manu Shantharam, Mary Hall, Michael Garland and Bryan Catanzaro, Nitro: A Framework for Adaptive Code Variant Tuning, 2014 IEEE 28th International Parallel and Distributed Processing Symposium, pp. 19-23 （2014）.
2) Hang Cui, Shoichi Hirasawa, Hiroyuki Takizawa and Hiroaki Kobayashi, A Code Selection Mechanism Using Deep Learning, 2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip, pp. 385-392 （2016）.
3) 自動ブロック化・通信最適化ライブラリ ABCLib Xabclib プロジェクト, http://www.abc-lib.org/Xabclib/index-j.html.
4) The SuiteSparse Matrix Collection, https://sparse.tamu.edu.
5) TensorFlow-an open-source software library for Machine Intelligence,https://www.tensorow.org/.
6) OpenCV, http://opencv.org.
7) The SuiteSparse Matrix Collection Rommes/xingo_afonso_itaipu, https://sparse.tamu.edu/Rommes/xingo_afonso_itaipu.
8) T. Katagiri, K. Kise, H. Honda, and T. Yuba, ABCLibScript: a directive to support specification of an auto-tuning facility for numerical software, Parallel Computing, Vol. 32, Issue 1, pp.92-112 (2006).
9) T. Katagiri, S. Ohshima, and M. Matsumoto, Directive-based auto-tuning for the finite difference method on the Xeon Phi, Proceedings of IEEE IPDPSW2015, pp. 1221–1230 (2015).
10) T. Katagiri, M. Matsumoto, and S. Ohshima, Auto-tuning of Hybrid MPI/OpenMP Execution with Code Selection by ppOpen-AT, Proceedings of IEEE IPDPSW2016, pp. 1488–1495 (2016).
11) T. Katagiri, S. Ohshima, and M. Matsumoto, Auto-tuning

ⓒ2017 Information Processing Society of Japan

on NUMA and Many-core Environments with an FDM Code, Proceedings of IEEE IPDPSW2017 (2017).

ⓒ2017 Information Processing Society of Japan