卒 業 研 究 報 告 書

題　目　　深層学習モデルにおける差分の
時系列変化可視化ツール

指導教員　　水野　修　教授

崔　恩瀞　助教

京都工芸繊維大学 工芸科学部 情報工学課程

学生番号　　17122014

氏　　名　　大橋　幸奈

令和3年2月12日提出

深層学習モデルにおける差分の時系列変化可視化ツール

令和 3 年 2 月 12 日　　17122014　大橋　幸奈

概　要

深層学習モデルとは，人間の脳の構造に着目した多層パーセプトロンを用いたプロセスである．深層学習モデルは，多様なパターンを学習することができるため，近年，自動運転や画像認識など様々な分野で応用されている．多様なパターンのデータを学習できる理由の一つに，モデルが調整できるパラメータを多く持つことがあげられる．

最適なモデルやパラメータの組み合わせを見つけるために，深層学習モデルを予測精度などにより評価し，これらのパラメータを何度も調整する試行錯誤が必要である．この試行錯誤によって，パラメータと精度の組み合わせのデータなども多く生まれる．しかし，こうした組み合わせなどのデータを適切に管理する方法はなく，開発者それぞれが自力で対処している．

本研究ではこの問題における解決策として，深層学習モデルの差分とその精度をグラフ化し，ある時点のモデルに至るまでの履歴をフロー図として可視化するツールを開発した．RNN や CNN などのモデルの構造や，データの前処理部分のパラメータ，層の数，層の中のパーセプトロンの数（ユニット数），また各パーセプトロンの繋がり方や出力の仕方などのハイパーパラメータを本研究における深層学習モデルにおける差分とする．このツールによって，開発者がモデルに何か変更を加えた時，変更したパラメータを含む差分と，精度を表計算ソフトなどに打ち込むなどの手間をかけることなくグラフ化できる．また，そのモデルに至るまでのコミットしてきたファイルの情報もフロー図として可視化できる．深層学習モデルのコミット間の差分とその精度を可視化することによって，パラメータ調整などの試行錯誤の履歴を，すぐに参照することができる．

目　次

1. 緒言

深層学習モデルとは，人間の脳の構造に着目した多層パーセプトロンを用いたプロセスである．深層学習モデルは，多様なパターンを学習することができるため，近年，自動運転や画像認識など様々な分野で応用されている．多様なパターンのデータを学習できる理由の一つに，モデルが調整できるパラメータを多く持つことがあげられる．

例えば，深層学習モデルを実際に適用する際のプロセスは大きく分けると，

1. 集めたデータを実際の学習に使えるように加工したり，人工的に数を増やしたりするデータの前処理
2. パターンを捉えるためのモデルの構築
3. モデルがパターンを見つけられるようにする学習
4. 最も良いモデルを選択するための性能評価，またモデルを利用しての予測や次元削減

という４つのステップとなる．

この時のパラメータとして，データの前処理部分におけるパラメータ，活性化関数，隠れ層の数，隠れ層のユニット数活性化関数，ドロップアウトする割合，学習率，最適化関数，誤差関数，バッチサイズ，エポック数，正則化などのハイパーパラメータなどがある．このようにモデルのパラメータは膨大である．

また，最適なモデルやパラメータの組み合わせを見つけるために，深層学習モデルを予測精度などにより評価し，これらのパラメータを何度も調整する試行錯誤が必要である．この試行錯誤によって，パラメータと精度の組み合わせのデータなども多く生まれる．

様々なモデルを構築して試行錯誤を繰り返すことによって，試したパラメータの組み合わせやモデルの精度の値など，多くの管理が必要なデータが生まれる．後々に，手作業でデータをまとめて可視化することもできるが，データの量が膨大になるにつれてデータをまとめる手間とコストは増えていってしまう．また，深層学習モデルの精度が極端に低い，高いまたは不連続に反復する，損失値が不安定または下がらない，オーバーフィット，勾配の爆発などの問題が起こったとき，原因となるパラ

メータを見つける必要がある．パラメータ変更の履歴と精度の値を管理ができていないと，原因となるパラメータを見つけ，修正する手間とコストも増えてしまう．

本研究ではこの問題における解決策として，深層学習モデルの差分とその精度をグラフ化し，ある時点のモデルに至るまでの履歴をフロー図として可視化するツールを開発した．

1. モデル（例：RNN(Recurrent Neural Network) や CNN (Convolutional Neural Network))の構造
2. データの前処理部分におけるパラメータ
3. ハイパーパラメータ（活性化関数，隠れ層の数，隠れ層のユニット数活性化関数，ドロップアウトする割合，学習率，最適化関数，誤差関数，バッチサイズ，エポック数）

以上のモデルに関するデータのコミット間の差を本研究の差分とする．開発者が Git のリポジトリにコミットする際に，コミットメッセージに決められたコマンドを書き，そのコミット情報および深層学習モデルのコミット間の差分を取得することによって可視化を実現している．このツールによって，開発者がモデルに何か変更を加えた時，変更したパラメータを含む差分と，精度を表計算ソフトなどに打ち込むなどの手間をかけることなくグラフ化できる．また，そのモデルに至るまでのコミットしてきたファイルの情報もフロー図として可視化できる．深層学習モデルのコミット間の差分とその精度を可視化することによって，パラメータ調整などの試行錯誤の履歴を，すぐに参照することができる．本論文では，可視化ツールの評価は行わないが，今後の課題としての評価の方法を提案している．

最後に本論文の以降の構成を示す．まず，第 2 章で本研究の必要となる背景知識について言及しながら，関連研究を紹介する．第 3 章では本研究の目的とその提案手法を述べる．第 4 章では 可視化ツールの仕様を述べるため，可視化の対象，フロー，コードを，順を追って説明する．第 5 章ではツールの利用シナリオについて述べる．最後に第 6 章で本研究の今後の課題であるツールの評価方法とを述べ，結言を述べる．

2. 背景

本章では，関連研究を紹介しながら，深層学習を含んだソフトウェアを開発するプロセスと課題を説明する．また実装の参考となった研究について解説する．

2.1 深層学習を用いたソフトウェアの開発のプロセス

深層学習モデルの実装は，Pytorch[1]，Keras[2]，tensorflow[3] など様々なフレームワークのおかげで，多くの開発者が比較的簡単に手を出すことができる [4].

一方で，深層学習モデルは第 1 章でも述べた通り，多くの調整が必要なパラメータを持っている．例えば，データの前処理部分におけるパラメータ，活性化関数，隠れ層の数，学習率，最適化関数，正則化 [5] などがある．これらのパラメータを最適に設定できないと精度が向上しなかったり，精度が極端に低い，高いまたは不連続に反復する，損失値が不安定または下がらない，オーバーフィット，勾配の爆発などの問題が起こる [6]．また，パラメータの構成が乱雑であると，それらの問題に対する原因を見つけるのが難しくなり，修正する手間とコストも増えてしまう [7][8]．パラメータ設定のミスはコストがかかり，深刻な時間の損失，計算の浪費につながる．

そのため，パラメータの調整は重要であり，そのパラメータの管理もまた重要である．パラメータを管理する方法として，構成設定の検証や，２つの構成設定間の差分をとったりすることが効果的である [9]．Zhang らは人気のある Q&A サイト Stack Overflow の深層学習の質問に対する大規模な実証研究を行い，ハイパーパラメータの選択のための，TensorBoard と Visdom が開発した全体のトレーニングプロセスを可視化ツールを紹介している [6].

2.2 可視化ツール

本研究では，深層学習のモデル間における差分を精度とともに可視化することで，パラメータを管理する．可視化ツールを実装するうえで，以下に示す研究のグラフの形式を参考にした．

Alcocer らは行列のレイアウトを使用して，各セルには，あるソフトウェアコンポーネント（行）のあるバージョン（列）での実行時のソースコードと実行時のメト

リクスを可視化している [10]．これにより，実務者は一度に複数のソフトウェアバージョンに沿ったソフトウェアコンポーネントの性能を分析することができる．また，マトリクスは複数のインタラクションをサポートしており，要求に応じて詳細な情報を表示することができる．

Tomida は，遺伝的アルゴリズムのつながりにヒントを得て，プログラムの自動修復処理におけるプログラムの進化を可視化するためのソフトウェアツールフォーク GenProg として実装した [11]．提案する可視化手法は，コードの系譜を鳥瞰図のためのツリー構造として表現した．

Evan らは，コミットがどのように Linux のマスターブランチにマージされるのかを示すマージツリーを提案した [12]．可視化ツールは，それぞれのマージツリー内のコミットやマージの位置，編集されたファイル，編集されたモジュール，コミットメッセージに関する情報を提供した．

2.3 問題提起

深層学習では，多くのモデルを作ることが可能であるが，それに伴って設定しなければいけないパラメータが多く存在する．また，最適なモデルやパラメータの組み合わせを見つけるために，これらのパラメータを何度も調整し，試行錯誤する必要がある．そしてこの試行錯誤によって，パラメータと精度の組み合わせのデータなども多く生まれるのでそのデータ管理が問題となる．

それらを可視化する上で，データをグラフで表すことは有効である．実際，Evan ら [12] のマージツリーは，マージ情報をグラフによって示している．しかし，既存の研究では，深層学習モデルが持つ多くのパラメータを可視化し，管理を用意にする手法は提案されていない．

3. 目的

3.1 本研究の目的

本研究の目的は，ユーザーが直観的かつ簡単に，深層学習モデルを用いたシステムの精度向上のための試行錯誤の履歴を見ることができる深層学習モデルにおけるデータの差分の時系列変化可視化ツールの開発である．データ管理の方法として，Git のコミットメッセージに書いたコマンドによって管理する方法を提案する．

3.2 提案手法

本研究では Git のコミットメッセージに，モデルの差分と精度に関するコマンドを書いておき，後からツールを立ち上げた時，Git のコミットログから情報を受け取って可視化する方法でツールを実装する．コミットメッセージの例：

- change param [差分を取るデータ名] [変更前の値] to [変更後の値] from [変更前の値を利用したファイル名] to [変更後の値を利用したファイル名]
- metrics [使用した評価指標] [評価指標の値] in [評価したファイル名]

コマンドが書かれたコミットのみを抽出するので，予測精度を出していないファイルや可視化したくないファイルは普通にコミットできる．

可視化には，ネットワークグラフとフロー図を使用する．ネットワークグラフとは，頂点（ノード）と辺（エッジ）によって構成されるグラフである．エッジがどのノードとノードを繋ぐのかを定義することによって，ネットワークグラフの構造は決まる．一回のコミットで，2 つのファイルを一緒にコミットした場合，グラフでは，同じ x 線上に 2 つのノードをプロットする．ネットワークグラフは，ただの折れ線グラフよりも自由度が高く，これらの場合に対応できるので今回はネットワークグラフを利用した．

グラフにプロットされたファイルがそのモデルに至るまでのデータの差分の履歴を分かりやすく可視化するためにフロー図を使用する．このフロー図には，詳しいコミット情報も表示されるので，ファイル名，コードを作った人，最初にコミットした日時，コミットした人，コミット変更日時，コミットメッセージに書かれたコマン

ド，変更されたパラメータ，パラメータの差分，変更元のファイル名，評価したメトリクス，評価した値なども確認できる．

4. 可視化ツールの仕様

本章では，深層学習モデルにおける差分の時系列変化可視化ツールの仕様について述べる．

最初にツールを立ち上げると，精度とコミット日時を軸として，コミットされたファイルをグラフ上に表示する（図 4.1）．次に，図 4.1 のプロットされた丸をクリックすると，図 4.2 のようなフロー図が表示される．これは図 4.1 の一番上の丸をクリックした時にそのノードの詳細情報と，そのノードに線で繋がっているノードの詳細情報が表示される．その時，図 4.3 のようにクリックされたノードのみがハイライトされるようになる．

4.1 可視化の対象

本研究では，深層学習モデルにおける差分の時系列変化可視化ツールを開発する．ここでの差分を取るデータとして以下を定義する．

1. モデル（例 : RNN(Recurrent Neural Network) や CNN (Convolutional Neural Network)）の構造
2. データの前処理部分におけるパラメータ
3. ハイパーパラメータ（活性化関数，隠れ層の数，隠れ層のユニット数活性化関数，ドロップアウトする割合，学習率，最適化関数，誤差関数，バッチサイズ，エポック数）

以上のモデルに関するデータのコミット間の差を本研究の差分とする．

4.2 可視化までのフロー

深層学習モデルにおける差分の時系列変化可視化ツールを利用するまでのフローを説明する．

1. 開発者が，差分を取るデータを指定する．
2. git commit 時に，開発者が差分と結果の精度部分を取り出しコミットメッセージにコマンドを書き込む．

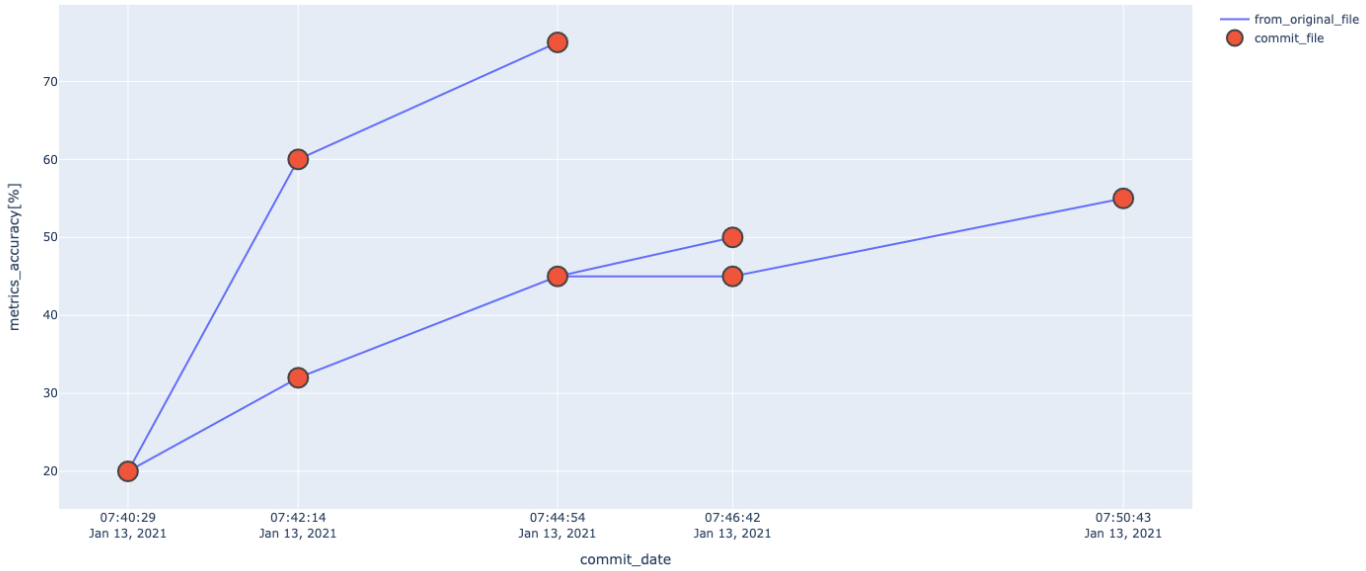

図 **4.1**　精度と時系列グラフの例


Change Log
File_name cifar10_01.ipynb
author_date 2021-01-13 07:44:54+09:00
author_name yukina ohashi
command change param height_shift_range 0.3 to 0.4 from cifar10_01.ipynb to cifar10_01.ipynb metrics accuracy 75% in cifar10_01.ipynb change param width_shift_range 0.3 to 0.4 from cifar10_02.ipynb to cifar10_02.ipynb metrics accuracy 45% in cifar10_02.ipynb
committer_date 2021-01-13 07:44:54+09:00
committer_name yukina ohashi
metrics accuracy
metrics_value 75%
original_file cifar10_01.ipynb
parameter height_shift_range
parameter_change 0.3 to 0.4
^
File_name cifar10_01.ipynb
author_date 2021-01-13 07:42:14+09:00
author_name yukina ohashi
command change param height_shift_range 0.2 to 0.3 from cifar10_01.ipynb to cifar10_01.ipynb metrics accuracy 60% in cifar10_01.ipynb change param width_shift_range 0.2 to 0.3 from cifar10_01.ipynb to cifar10_02.ipynb metrics accuracy 32% in cifar10_02.ipynb
committer_date 2021-01-13 07:42:14+09:00
committer_name yukina ohashi
metrics accuracy
metrics_value 60%
original_file cifar10_01.ipynb
parameter height_shift_range
parameter_change 0.2 to 0.3
^
File_name cifar10_01.ipynb
author_date 2021-01-13 07:40:29+09:00
author_name yukina ohashi
command metrics accuracy 20% in cifar10_01.ipynb
committer_date 2021-01-13 07:40:29+09:00
committer_name yukina ohashi
metrics accuracy
metrics_value 20%
original_file
parameter
parameter_change

図 4.2　フロー図の例

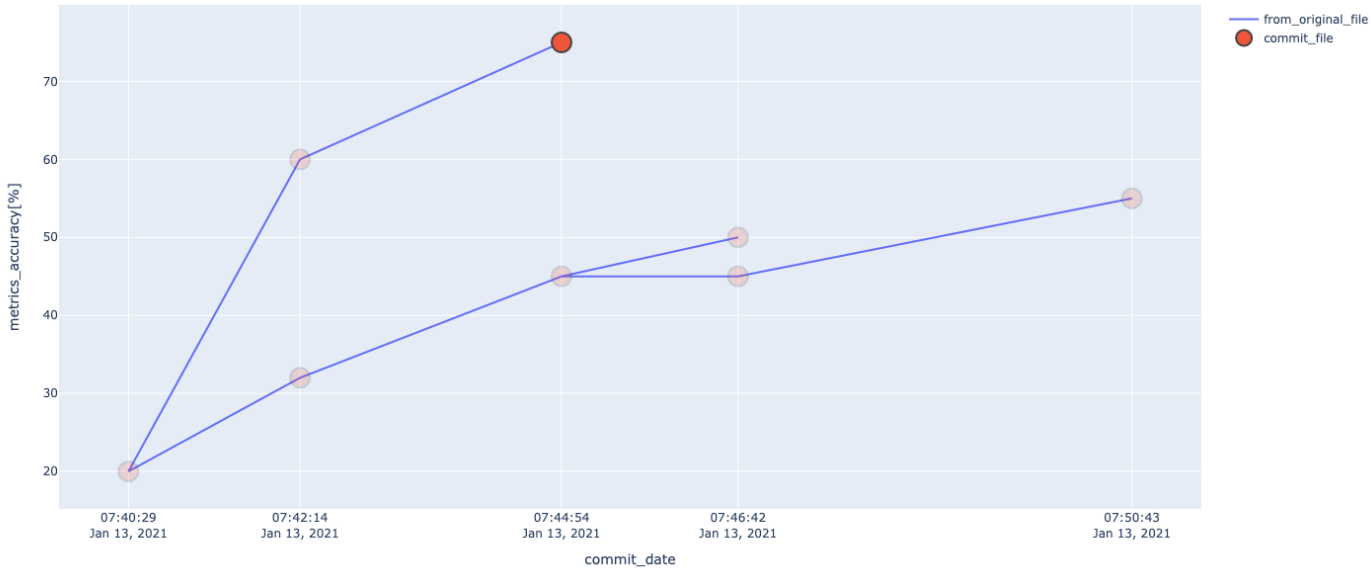

図 **4.3**　ノードクリック時の精度と時系列グラフの例

- change param [差分を取るデータ名] [変更前の値] to [変更後の値] from [変更前の値を利用したファイル名] to [変更後の値を利用したファイル名]
- metrics [使用した評価指標] [評価指標の値] in [評価したファイル名]

ここでいう，差分を取るデータ名とは 4.1 節で示したモデルやパラメータ名を指す．変更前と変更後の値とは，例えばあるパラメータを 0.1 から 0.2 へ変更した場合，変更前の値は 0.1，変更後の値とは 0.2 を指す．変更前の値を利用したファイル名とは，前の例で 0.1 のパラメータを利用したモデルのファイル名，変更後の値を利用したファイル名とは，前の例で 0.2 のパラメータを利用したモデルのファイル名，つまりその時コミットしているファイル名ということになる．使用した評価指標名とは，テスト用データセットに対する予測精度の評価指標を指す．評価したファイル名とは，前の評価指標を適用したモデルのファイル名を指す．

3. Git のローカルリポジトリの内容をリモートリポジトリに送信する．
4. ツールを利用すると，新しく追加されたコミットの情報と差分の情報，精度の情報が取得され可視化される．

コミットメッセージにコマンドが書かれていない場合には，そのコミットは無視されるので，コミットごとに学習をし精度を出す必要はない．

4.3 実装

Python と JavaScript を用いて，時系列と精度のグラフ，コミット情報のフロー図を html で表示するプログラムを作成した．このプログラムは，

1. コミット情報を json ファイルにまとめる
2. json ファイルからグラフにプロット
3. プロットされたファイルをクリックすると現れるフロー図を表示

以上の流れで処理している．それぞれの処理について以下で順に詳しく説明する．

4.3.1 コミット情報を json ファイルにまとめる

Git のコミットログから情報を取得して，それぞれのコミットで，変更後のファイルごとに json 形式でファイルに情報をまとめる．一つの変更後のファイルはそれぞれ一つの予測モデルとする．リポジトリにコミットされる際に，コマンドで指定された変更後のファイルのみ，json 形式としてファイルに書きだす．例えば，コミットするファイル名を foo.ipynb とする．精度の指標を正解率（accuracy）として，その値が 60 %である場合，コミットメッセージに

metrics accuracy 60 % in foo.ipynb

とコマンドを書く．ここで，指定された変更後のファイルのコミット情報を取得する．コミット情報とは，ファイル名，コードを作った人，最初にコミットした日時，コミットした人，コミット変更日時，コミットメッセージに書かれたコマンド，変更されたパラメータ，パラメータの差分，変更元のファイル名，評価したメトリクス，評価した値，である．ただし，コマンドによる入力がない場合は，ヌル文字を渡す．

また，コマンドに書かれた情報も一緒に json 形式でファイルに書きだす．前のコミットからの差分に関するコマンドが，コミットメッセージとして書かれている場合も一緒に json 形式でファイルに書きだす．

例えば，変更前のファイル名，foo.ipynb で，height-shift-range というパラメータが 0.1 という値を取り，次のコミットでその値を 0.2 に変更してファイル名は変更せずコミットした場合，コミットメッセージにはコマンドとして，

change param height-shigt-range 0.1 to 0.2 from foo.ipynb to foo.ipynb

と書かれ，その情報も json 形式でファイルに一緒に書き出す．

入力 : Git のディレクトリ名をコマンドラインから与える．Python でコマンドを扱える subprocess モジュールを使って，コミットログからコミット情報を取得する．

出力 : 精度に関するコマンドがコミットメッセージに書かれているコミットのハッシュ値をキーとして取得．精度に関するコマンドに書かれているファイルを値として取得．そのファイルをキーとして，コミット情報とコマンド情報を値として取得する．

4.3.2 精度とコミット時間のグラフ

json ファイルを読み込み，ネットワークグラフを作成する．json ファイルに，前にコミットした時からの差分が書かれていた場合，前にコミットしたファイルと，変更後のファイルのプロットされたノード同士が線で繋がれる．例えば，変更前のファイル名，foo.ipynb で，height-shift-range というパラメータが 0.1 という値を取り，次のコミットでその値を 0.2 に変更してファイル名は変更せずコミットした場合，コミットメッセージには，

change param height-shigt-range 0.1 to 0.2 from foo.ipynb to foo.ipynb

と書かれ，前にコミットした foo.ipynb ファイルと後にコミットした foo.ipynb ファイルのノードが線で結ばれる．

入力 : Git のディレクトリ内にある commit-log.json という名前の json 形式のファイルを読み込む．キーのハッシュ値ごとに入っているファイルの一つがグラフのノード一つに対応する．ノード情報として，変更元のファイルのノードの情報を入れる．それを元にしてノード同士の辺（エッジ）をグラフに足していく．また，ノードクリック時のイベントを制御する js ファイルと，html のフロー図部分のデザインをする css ファイルを読み込む．

出力 : x 軸にコミットされた時刻，y 軸にモデルの精度をとって，ネットワークグラフにノードとエッジをプロットする．ネットワークグラフの図を html として出力させ，ブラウザで表示してノードにカーソルを合わせると，ポップアップにハッシュ値とファイル名を合わせた情報を表示する．ノードをクリックしたとき，そのノードのみがハイライトされるようになり，クリック時に JavaScript で扱えるイベントを発行する．

4.3.3 フロー図

ネットワークグラフにプロットされたノードをクリックした時のイベントを処理する．ファイルのコミット情報と変更前のファイルのコミット情報をフロー図で表示する．ネットワークグラフにプロットされたノードをクリックすると，対応するファイルのコミット情報が表示される．そのファイルのノードが，別のファイルのノードと線で繋がっていた場合，コミットされた時刻の順番にコミット情報が表示される．

入力：イベントに関連するノードのデータはイベント発行元の py ファイルから取得される．データには，選択されたノードの情報が含まれており，そこから変更元のファイルのノード情報などのデータを取得できる．

出力：DictToFlow() という関数から，ファイルのデータを DictToTable() という関数に渡す．返り値として受け取った table 要素を lu 要素の子要素として追加する．また，lu 要素を div 要素の子要素として html のなかに追加する．

5. 利用シナリオ

ある会社の開発チームとしてAさんは深層学習を用いて，画像認識システムを作っていた．深層学習用のライブラリTensorFlowの中の’tensorflow.keras’モジュールを利用することにした．CNNモデルを用いることに決め，大きなズレや回転，反転などに対しては同じ特徴だと認識させるために，学習データに人工的にさまざまな変換を加えデータを増やすことにした．kerasの実装においてはkeras.preprocessing.imageを用いて，元の画像を上下にずらすことにした．

height-shift-range=0.3

と最初のパラメータを設定し，学習させるとaccuracyは60 %とでた．次に，

height-shift-range=0.4

と最初のパラメータを調整し，学習させるとaccuracyは70 %とでた．同じようなパラメータ調整を10回以上行った．その時点で，Aさんは別のプロジェクトに移動しなければならなくなり，画像認識システムはBさんに引き継ぐことになった．Aさんは，今までのパラメータ調整の履歴をBさんに理解してもらう必要があった．なので今までの10回のパラメータ変更の履歴とその予測精度をコードを見ながら確認し，表計算ソフトを使って見やすいグラフにした．それをBさんに見せ，仕事を引き継ぐことができた．

Bさんは引き継いだ深層学習モデルの精度をもっとあげるため，モデルのハイパーパラメータの学習率を調整することにした．この時，記録しておきたいデータを持つモデルと対応するファイルをコミットする際に，

change param lr 0.01 to 0.02 from foo.ipynb to foo.ipynb

metrics accuracy 80 % in foo.ipynb

というコマンドをコミットメッセージに書き加えてからコミットした．同じようなパラメータ調整を10回以上行った．その時点で，Bさんは別のプロジェクトに移動しなければならなくなり，画像認識システムはCさんに引き継ぐことになった．Bさんは，今までのパラメータ調整の履歴をCさんに理解してもらう必要があった．Bさんは，本研究の深層学習モデルにおける差分の時系列変化可視化ツールを立ち上げ，今までのパラメータ調整を可視化した．グラフを見せながら，Cさんに学習率の変更による精度向上の履歴を伝えた．その後，Cさんは速やかにBさんの仕事

を引き継ぐことができた．

この例では、A さんから B さんの引き継ぎ時に、手作業での確認と，グラフ化するツールの使用があり，引き継ぎ作業が煩雑である．一方，提案したツールを使った B さんから C さんの引き継ぎでは，この手間を省略できる．

6. 結言

6.1 本研究の貢献

本研究では，深層学習モデルを含むソフトウェアの開発における，パラメータの値と精度の関係などデータ管理のための，データの差分の時系列変化可視化ツールを開発した．具体的には，精度とコミット日時を軸として，コミットされたファイルをノードとし，その関連をエッジとして表したネットワークグラフとして可視化する．さらに，ノードをクリックするとそのファイルに至るまでのファイルのコミット情報をフロー図によって可視化する．可視化によって，深層学習のモデルを開発する際のパラメータ調整などの試行錯誤の差分を精度とともに，参照でき，引き継ぎなどで開発者を支援することができる．

6.2 今後の課題

今後の課題として，本研究で開発した深層学習モデルにおける差分の時系列変化可視化ツールの評価を行い，その有効性を示す必要がある．本章では可視化ツールをユーザビリティ評価する方法について提案する．

6.2.1 実験概要

実装した深層学習モデルにおける差分の時系列変化可視化ツールに対して，被験者による評価実験を行う．この実験では被験者に満足度を評価してもらうことによってツールの有効性を示す．被験者には，利用シナリオに沿ったタスクをツールがある場合とない場合で行ってもらい，その後，ツールに対する満足度アンケートを行う．

6.2.2 タスク

被験者には以下のタスクを与え，深層学習モデルにおける差分の時系列変化可視化ツールを評価してもらう．深層学習を用いたシステムのモデルのファイル foo.ipynb と bar.ipynb を被験者に提示する．

1. 被験者に foo.ipynb を提示する．

2. 被験者に自由に 10 回パラメータを調整してもらってできる限り高い予測精度を目指してもらう.
3. 最後にどのパラメータの組が最も良かったかを報告してもらう.
4. 被験者に bar.ipynb を提示する.
5. 被験者に自由に 10 回パラメータを調整してもらってできる限り高い予測精度を目指してもらう．この時，各調整時にコミットメッセージとしてツールに必要なコマンドを入力してもらう.
6. 最後にどのパラメータの組が最も良かったかを報告してもらう.
7. 最後に，ツールを使った場合と使っていなかった場合の満足度についてアンケートを取る.

6.2.3 評価指標

評価指標としては，アンケートによる満足度評価を用いる．ツールは html を実行するため，ウェブユーザビリティ評価スケールを用いて，ツールの評価項目として，WUS[13] の 7 つの評価因子

1. 内容の信頼性
2. 役立ち感
3. 操作のわかりやすさ
4. 構成のわかりやすさ
5. 見やすさ
6. 反応のよさ
7. 好感度

を設定する．項目ごとに 5 段階評価を行う.

謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢，本報告書の作成に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学・人間科学系 水野修教授，崔恩瀞助教, 名古屋大学 大学院情報学研究科 吉田 則裕准教授に厚く御礼

申し上げます. 本報告書執筆にあたり執筆，校閲等 全面的に協力頂きました本学設計工学専攻 近藤将成先輩，全般にわたって貴重な助言を多数いただきましたソフトウェア工学研究室の皆さん， 学生生活を通じて著者の支えとなった家族や友人に深く感謝致します.

参考文献

[1] Facebook,Inc., PyTorch, （オンライン）, 入手先〈https://pytorch.org〉（参照 2021-2-10）.

[2] Keras:Python の深層学習ライブラリ,（オンライン），入手先〈https://keras.io/ja/〉（参照 2021-2-10）.

[3] Google,Inc.，TensorFlow を選ぶ理由，（オンライン），入手先〈https://www.tensorflow.org/?hl=ja〉（参照 2021-2-10）.

[4] Q. Guo, S. Chen, X. Xie, L. Ma, Q. Hu, H. Liu, Y. Liu, J. Zhao, and X. Li, "An empirical study towards characterizing deep learning development and deployment across different frameworks and platforms," In Proceedings of the 34th International Conference on Automated Software Engineering, p.810–822, Nov. 2019.

[5] AlexandraL'Heureux，K. Grolinger，H.F. Elyamany，M.A.M. Capretz，"Machine learning with big data: Challenges and approaches," IEEE Access, pp.7776–7797, Feb. 2017.

[6] T. Zhang, C. Gao, L. Ma, M. Lyu, and M. Kim, "An empirical study of common challenges in developing deep learning applications," In Proceedings of the 30th International Symposium on Software Reliability Engineering, pp.104–115, Oct. 2019.

[7] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning:a case study," In Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, p.291–300, May 2019.

[8] L. Berti-Équille, A. Bonifati, and T. Milo, "Machine learning to data management: A round trip," In Proceedings of the 34th International Conference on Data Engineering, pp.1735–1738, April 2018.

[9] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, "Hidden technical debt in machine learning system," In Proceedings of the 28th International Conference on Neural Information Processing Systems, p.2503–2511, Dec. 2015.

[10] J.P.S. Alcocer, F. Beck, and A. Bergel, “Performance evolution matrix: Visualizingperformance variations along software versions,” In Proceedings of the seventh Working Conference on Software Visualization, pp.1–11, Sept. 2019.

[11] Y. Tomida，Y. Higo，S. Matsumoto, and S. Kusumoto, “Visualizing code genealogy — how code is evolutionarily fixed in program repair? —,” In Proceedings of the seventh Working Conference on Software Visualization，pp.23–27，Sept. 2019.

[12] E. Wilde and D. German, “Merge-tree: Visualizing the integration of commits into linux,” In Proceedings of fourth Working Conference on Software Visualization, pp.1–10, Oct. 2016.

[13] 株式会社イード — すべての人に最高のユーザーエクスペリエンスを !,（オンライン），入手先〈https://www.iid.co.jp〉（参照 2021-2-10）.