ディープラーニングを用いた 学習エージェントの開発支援機構の構築

渡邉 賢人[1] 打矢 隆弘[1] 内匠 逸[1] 木下 哲男[2]

概要：近年多様化するニーズに有効な手法として，エージェント指向コンピューティングが注目されている．本研究で取り扱うエージェントフレームワーク DASH では，学習性を備えたエージェントの開発は可能であるが，現在の開発環境では画像認識や音声認識といった幅広い用途への応用は困難である．これらの応用が可能な学習の手法としてディープラーニングが挙げられる．しかし，ディープラーニングを利用するためには詳しい知識を必要とし，一般のエージェント開発者が利用することは困難である．そこで，本研究ではディープラーニングによる多様な学習が可能なエージェントを提案するとともに，詳しい知識を持たない開発者による開発を支援することで学習エージェントの容易な開発を可能とする．

Construction of Development Support Mechanism for Deep Learning Agent

KENTO WATANABE[1] TAKAHIRO UCHIYA[1] ICHI TAKUMI[1] TETSUO KINOSHITA[2]

1. はじめに

近年，インターネットの急速な普及に伴い，ネットワークサービスは大きな発展を遂げている．これにより，ユーザの要求は多様化し，様々なニーズに柔軟に対応できるシステムの必要性が高まっている．このようなシステムを実現する手段として，エージェント指向コンピューティングに注目が集まっている．

エージェントとは，ユーザがある目的を達成するためにユーザの代わりに動作するソフトウェアであり，ユーザや他のソフトウェアと通信しながら自律的に動作する．エージェントは，エージェントフレームワークと呼ばれる枠組みにより，効率的に管理・運用を行うことが可能である．エージェントには「学習性」と呼ばれる特性を付与することができる．学習性とは，過去の経験から最適な行動を学習する特性である．学習性を有するエージェントを学習エージェントと呼ぶ．

従来のエージェントフレームワークでは，学習エージェントの開発は可能である．しかし，学習エージェントを開発するためには開発者が学習性を実装するコードを一から記述する必要がある．エージェントに対する学習性の実装を支援するための先行研究として “知的マルチエージェントシステムにおける強化学習エージェント設計支援機構” [1] がある．これはエージェントフレームワーク DASH[2] を対象としており，エージェントが行動を選択するためのルールに対し，新たに優先度を導入し，強化学習により優先度を更新する自動学習機能を組み込んでいる．そのため，複雑なコードを記述することなく学習性を実装可能である．しかし，画像認識や音声認識といった機械学習を行うエージェントを開発することはできないため，エージェントが解決できる問題は限られる．

本研究では，エージェントフレームワーク DASH を対象に，学習手法の一つであるディープラーニングを追加することを提案する．ディープラーニングとは，ニューラルネットワークを用いた機械学習の手法であり，画像認識，音声認識といった幅広い分野に応用されている．

また，ディープラーニングの詳しい知識を持たない開発者がディープラーニングを搭載したエージェントを開発するために，学習エージェントの開発支援を行う．これにより開発者の負担軽減や学習エージェントの開発時間の短縮を実現する．

2. エージェントフレームワーク DASH

DASH(Distributed Agent System based on Hybrid architecture) とは，エージェントを集積・管理するためのリポジトリとエージェントが生成され実際に動作するワークプレースにより構成されるエージェントフレームワークである．DASH はリポジトリ型マルチエージェントフレームワークと呼ばれる（図 1）．リポジトリでエージェント群を管理することにより，まずユーザがワークプレースに対してサービスを要求する．次にワークプレースがリポジトリに対してタスクを通知する．するとエージェントによるシ

1 名古屋工業大学 大学院工学研究科 情報工学専攻
2 東北大学 電気通信研究所

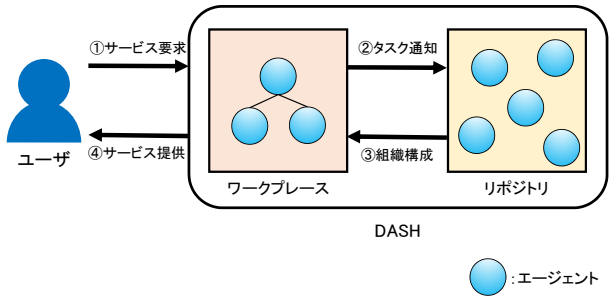

図 1 DASH によるサービス提供

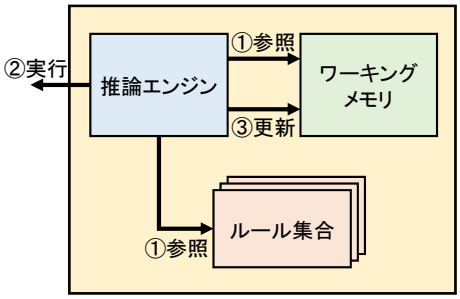

図 2 DASH エージェントの推論モジュール

ステムがワークプレース上に生成され，動作を行う．これにより，ワークプレース上のエージェント群がユーザに対してサービスを提供する．

2.1 DASH エージェント

エージェントフレームワーク DASH により作成されたエージェントを DASH エージェントと呼ぶ．DASH エージェントは，テキストファイルにより記述されている．DASH エージェントは，自身の行動を決定するための機構として推論モジュールを持つ（図 2）．推論モジュールは，推論エンジン，ワーキングメモリ，ルール集合により構成される．

推論エンジンはワーキングメモリの状態を参照し，ルール集合から動作知識を制御する．これにより，エージェントは動作を行う．ワーキングメモリには，エージェントが得た情報が格納される．この情報はファクトと呼ばれ，ファクトは OAV 型データと呼ばれる形式で保存される．OAV 型データとは，オブジェト (Object)，属性 (Attribute)，属性値 (Value) の組で表されるデータである．ルール集合には，if-then 型のルールで表される動作知識が格納されている．

DASH エージェントは次のように動作する．

(1) エージェントにメッセージが到着すると，推論エンジンがワーキングメモリおよびルール集合を参照する．
(2) 推論エンジンはワーキングメモリの内容にマッチするルールを探索し，マッチしたルールの実行部に記述されているエージェント動作アクションを実行する．
(3) 実行したルールによりワーキングメモリの内容が更新された場合，推論エンジンはワーキングメモリを参照し，内容を更新する．更新されなかった場合，ワーキングメモリは参照されない．

以上の繰り返しにより，DASH エージェントは動作を行う．

2.2 エージェント記述方式

DASH では，エージェントファイル（*.dash），ベースプロセス（*.class），ルールセット記述ファイル（*.rset）の 3 つのファイルを記述することで，DASH エージェントを作成する．

知識記述ファイル

知識記述ファイルでは，エージェントの動作知識を記述する．知識記述ファイルは以下の 5 種類の記述により構成される．

- property
  エージェントの起動時にファクトとしてワーキングメモリに格納される OAV 型データを記述する．ここで記述されたファクトは通常のファクトとは異なり，値の変更や削除といった操作を行うことができない．
- initial_facts
  エージェントの起動時にファクトとしてワーキングメモリに格納される OAV 型データを記述する．ここで記述されたファクトは通常のファクトと同様，値の変更や削除といった操作を行うことができる．
- include
  外部に保存したルールセット記述ファイルからルールセットを読み込むために，ルールセット記述ファイルのファイル名や，読み込むための条件を記述する．
- knowledge
  エージェントが持つ動作知識を if-then 型のルール形式で記述する．
- rule
  条件部とアクション部から成り，条件部には条件を，アクション部には条件部に記述した条件をマッチした時に実行する命令 (アクション) を記述する．

ベースプロセス

ベースプロセスでは，知識記述ファイルで用いるルール型の言語では記述できない処理を記述する．ベースプロセスは Java 言語を用いて記述されており，主に I/O 処理や GUI の表示に用いられる．DASH エージェントは，アクション部よりベースプロセスのメソッドを呼び出し，動作させる．

ルールセット記述ファイル

ルールセット記述ファイルでは，エージェントが利用するルールセットを記述する．知識記述ファイルに記述されたルールセット以外にも include 文でルールセット記述ファイルを呼び出すことにより，記述されたルールセットを利用することが可能となる．

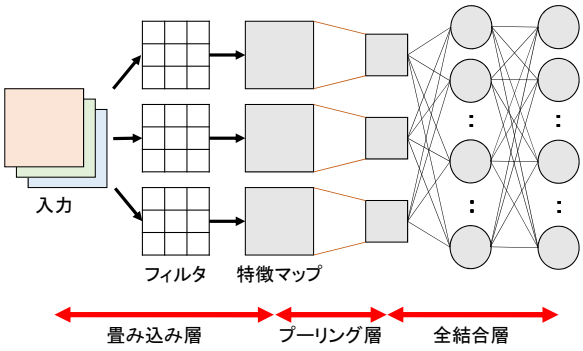

図 3　畳み込みニューラルネットワーク

3. ディープラーニング

ディープラーニングとは多層構造のニューラルネットを用いた機械学習の手法であり，深層学習とも呼ばれる．ディープラーニングは画像認識や音声認識といった多方面に応用でき，高い認識の精度を示すため，機械学習の手法として優れている．提案機構ではディープラーニングの手法として畳み込みニューラルネットワーク (Convolutional Neural Network) を用いる．

3.1 畳み込みニューラルネットワーク

畳み込みニューラルネットワークとは，図 3 のように局所的に結合し特徴を抽出する畳み込み層と，微小な位置の変化を吸収するプーリング層から構成される．畳み込みニューラルネットワークは画像認識の分野において優れた精度を誇り，現在のディープラーニング技術の主流と言える．

畳み込み層では近接する入力ノードをフィルタにより畳み込み演算を行い，特徴マップを出力する．特徴マップでは入力画像のエッジといった局所的な特徴を表現する．フィルタの値は誤差逆伝播法によって学習する，

プーリング層では近接する入力ノードのうち，最も大きい値のみを採用することで画像の圧縮を行う．これにより微小な位置の変化に対する不変性を得られるほか，次元を削減することで計算量を減らすことができる．

畳み込み層とプーリング層の後には，識別を行うための全結合層を繋げる．結合間の重みは誤差逆伝播法により学習する．

3.2 問題点

ディープラーニングは機械学習の手法として優れているが，アルゴリズムが複雑であること，設定するパラメータが多様であることといった問題点が存在する．よってディープラーニングを利用するためには詳しい知識が必要である．

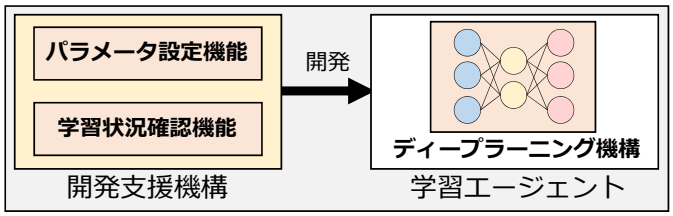

図 4　提案機構の全体図

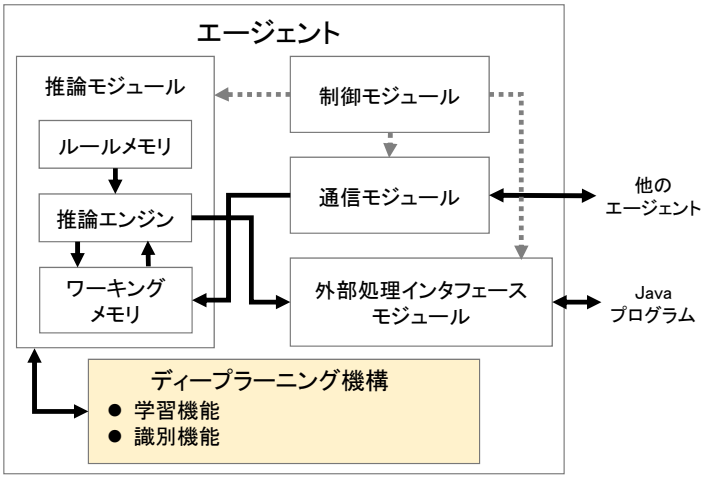

図 5　ディープラーニング機構を有するエージェント

4. 提案機構

提案機構は図 4 に示すように，エージェントがディープラーニングを行うディープラーニング機構とフレームワークからパラメータ設定や学習状況の確認を支援する開発支援機構により構成される．

提案手法の目的は，ディープラーニング機能を搭載するエージェントの開発に際しエージェント開発者の支援を行うことである．ディープラーニングはアルゴリズムが難解であり，設定するパラメータが多様であるため，専門の知識がないエージェント開発者がディープラーニングを用いたエージェントを開発することは困難である．提案手法ではフレームワークからディープラーニング学習エージェントの開発・運用を支援し，エージェントにディープラーニング機構を搭載する．これにより，エージェント開発者の負担を軽減するとともにエージェント上でディープラーニングを利用可能とする．

(S1) ディープラーニングアクションの追加

ディープラーニング機構は図 5 に示すように，推論モジュール全体と相互に作用する．推論モジュール内の推論エンジンよりディープラーニング機構を呼び出し，ディープラーニング機構の動作結果は推論モジュール内のワーキングメモリに保存される．

ディープラーニング機能は，学習機能と識別機能の 2 つに分けられる．これらの機能の動作について説明する．

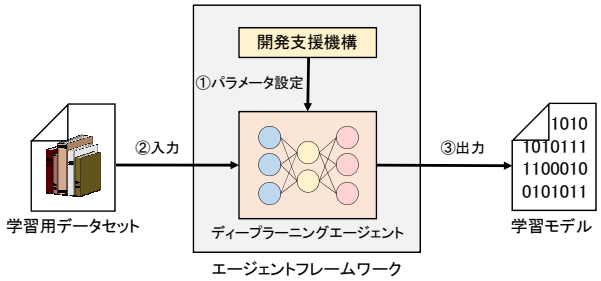

図 6 ディープラーニング機能：学習段階

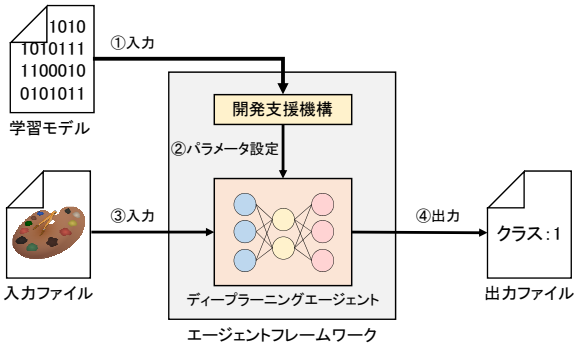

図 7 ディープラーニング機能：識別段階

学習機能

学習機能とは，提案機構が持つニューラルネットワークを学習する機能である．学習の流れについて説明する (図 6)．学習では，エージェントフレームワーク内でパラメータ設定したエージェントに対し，専用のフォーマットで記述された学習用データセットを入力することで学習を行う．学習用データセットは，入力データとそれが属するクラスを示すラベルが記述されており，提案機構により生成することが可能である．学習終了後，エージェントは学習結果を学習モデルとして外部へ保存する．学習モデルは，ディープラーニング学習エージェントが読み込むことで，後述の識別機能を用いることが可能となる．学習機能は，エージェントが training アクションを実行することで利用できる．記述例を以下に示す．

training アクションの記述例

```
(training "dataset.dat" "parameter.dat")
```

ここで dataset.dat とは，学習用データセットであり，DASH フレームワークと同じフォルダ以下に置いておく必要がある．また，parameter.dat はパラメータ設定ファイルであり，省略可能である．パラメータ設定ファイルとは，ディープラーニング機能のパラメータを設定するためのファイルであり，5.2.1 節のパラメータ設定機能で詳しく説明する．

識別機能

識別機能とは，ディープラーニング機能に対して入力を与え，その入力を分類する機能である．識別の流れを図 7 に示す．識別では，まず学習機能により生成された学習モデルを開発支援機構により読み込む．次に開発支援機構により読み込んだ学習モデルを備えたディープラーニングエージェントを開発し，専用のフォーマットで記述された入力ファイルを入力することで識別を行う．入力ファイルはエージェントが 1 個ずつ入力し，その都度出力を求める．入力ファイルは，提案機構で用いるために画像や音声といった様々なデータを変換したファイルであり，開発者により入力したい画像や音声といったデータをニューラルネットワークに入力できる 0 と 1 のみの形式に変換する必要がある．識別機能は，エージェントが classify アクションを実行することで利用できる．記述例を以下に示す．

classify アクションの記述例

```
(classify "input.dat" "abc")
```

ここで input.dat とは，入力ファイルであり，DASH フレームワークと同じフォルダ以下に置いておく必要がある．また，abc とはこの識別時に付加するタグであり，結果と合わせて出力ファイルに保存される．タグは開発者が任意に付加することができ，複数識別を行ったときに他の識別結果と区別するために利用できる．このタグを用いて識別結果の検索が可能となる．識別終了後，エージェントは識別結果を出力ファイルとして外部へ保存する．出力ファイルには今までの識別結果が全て記されている．出力ファイルの例を以下に示す．

出力ファイルの例

```
No:1,result:2,time:2015/1/1/15:34
No:2,result:0,time:2015/1/1/15:35,tag:abc
No:3,result:3,time:2015/1/1/15:38,tag:def
```

1 回の識別に対し 1 行で結果が保存され，識別結果は通し番号 (No)，結果 (result)，識別時刻 (time) が記述される．また識別時にタグを記述している場合，tag の欄に書き込まれる．結果は，提案機構で用意したベースプロセスにより，通し番号またはタグによる検索が可能である．この例では，1 個目の入力ファイルに対する識別ではクラス 2 に分類されている．2 個目の入力ファイルに対する識別ではクラス 0 に分類されており，かつ abc という文字列がタグとして識別時にエージェントにより付加されている．

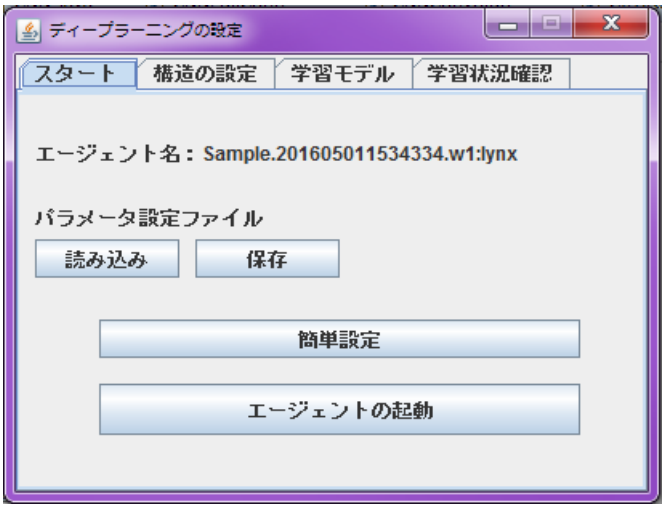

図 8 パラメータ設定画面

(S2) パラメータ設定機能を追加

ディープラーニングでは，設定する必要があるパラメータが多様であるという問題が存在していた．提案機構では，多様なパラメータを GUI 上で簡易に設定するための機能を実装している．ディープラーニング学習エージェントは，起動時にパラメータを設定するための GUI が表示される．この GUI を図 8 に示す．必要なパラメータの設定，学習モデルの読み込み・保存の設定はここで行われる．ここで，パラメータ設定画面上の各タブで設定できるパラメータについて説明する．

スタートタブ

最初に表示されるタブであり，パラメータ設定完了後，このタブよりエージェントを起動する．このタブより，設定されたパラメータをファイルに保存することが可能である．保存されたファイルは，読み込むことでパラメータ設定を行える．

構造の設定タブ

畳み込みニューラルネットワークの構造に関するを設定を行う．各層の数，ノード数のほかに学習回数や学習率といったパラメータをこのタブで設定する．

学習モデルタブ

学習モデルに関する設定を行う．学習結果の保存先の設定，すでに保存されている学習モデルの読み込みの設定を行う．

学習状況確認タブ

学習状況確認機能に関する設定を行う．このタブ上で，学習状況確認機能を利用するための設定を行う．学習状況確認機能については，(S3) 節で詳しく説明する．

(S3) 学習状況確認機能を追加

学習状況確認機能とは，学習中に学習の進度，経過時間，推定の残り時間，現在の学習精度といった学習の状況を図 9 に示す GUI 上で確認できる機能である．

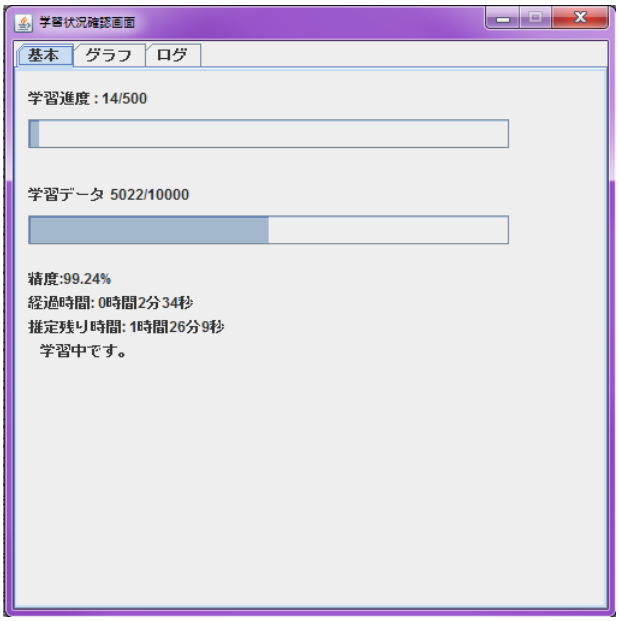

図 9 学習状況確認画面：基本タブ

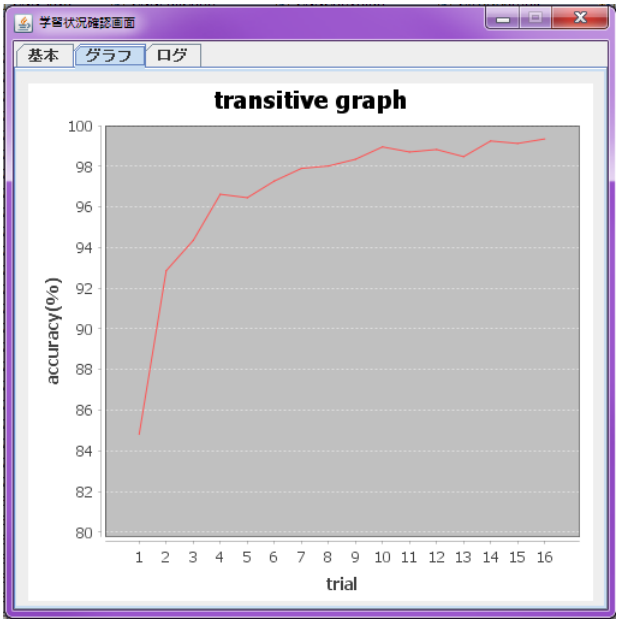

図 10 学習状況確認画面：グラフタブ

学習精度とは，学習されたニューラルネットワークに対して学習データを識別し，どれぐらい正しく識別できたかを表す割合である．学習状況確認画面では，この学習精度の推移を視覚的に確認するためのグラフ描画機能を利用可能である．グラフ描画機能の GUI を図 10 に示す．また，学習精度の推移はログとしてテキストで保存され，ファイルに保存される．

5. 実験と考察

本節では，実装した提案機構を用いた実験と考察について述べる．

5.1 実験環境

実験に用いた環境を表 1 に示す.

表 1 実験環境

OS	Windows 7 Professional
CPU	Intel(R)Core(TM)i7-4790 CPU@3.60GHz
メモリ	8.00 GB
システム種類	64bit OS

5.2 実験 1：ディープラーニング機能の有効性の検証

実装したディープラーニング機能により，適切に学習が行われることを確認するために実験を行う．実験では MNIST[3] を用いた画像認識を行う．MNIST とは手書き文字認識を行うためのデータセットであり，機械学習の評価を行うために利用されている．MNIST は 1 つずつ切り出された「0」～「9」の数字から構成されており，学習用に 60000 個，テスト用に 10000 個のデータを含んでいる．1 個のデータにつき，28×28 点の画像データと書かれている数字を表すラベルデータを持つ．画像データは $28 \times 28 = 784$ 点の各座標の明暗が 256 段階で記されており，0 が黒，255 が白となる．ラベルデータは画像データが「0」～「9」のどれかを表す 0～9 の数値が記されている．

実験の手順としては提案機構を用いて MNIST による学習を行う．この時，学習に用いるデータ数は 10000 個の場合と 60000 個の場合の 2 通りで行う．学習されたモデルは識別用のエージェントに搭載し，MNIST のテストデータ 10000 個を識別し出力ファイルを生成する．出力ファイルを確認し，正しく識別が行われた割合を認識率とする．

実験結果を表 2 に示す．結果より，学習によって十分な認識率が出ているため，ディープラーニング機能が正しく機能していることが確認できた．

表 2 学習データに対する比較結果

学習データ数	認識率 [%]	処理時間 [s]
10000	96.67	484
60000	98.31	932

5.3 実験 2：必要な開発ステップ数の調査

学習エージェント開発に必要な操作数を調査し，開発者の負担軽減について検証した．実験の手順として，まず提案機構を用いて学習エージェントを開発する際，以下の A～C の各操作を 1 ステップと定義する．そして学習エージェントが学習を行うまでのステップ数，および識別エージェントが識別を行うまでのステップ数を調査する．ステップ数の計測は DASH フレームワークを起動した直後から開始とし，学習エージェントは training アクション，識別エージェントは classify アクションの実行を目標とする．

- A : エージェントのコードを記述する (1 行あたり 1 ステップとする)
- B : GUI 上のボタン，チェックボックスを押す
- C : GUI 上のテキストボックスに文字を入力する (1 回の入力を 1 ステップとする)

調査したステップ数を表 3 に示す．結果より，十分少ないステップ数で学習エージェントを開発可能であることを確認できた．よって開発者によるコード記述，パラメータ設定の負担の軽減を実現していると言える．すなわち，本研究の目的である学習エージェントの開発支援という点において，提案機構は有効であると言える．

表 3 開発ステップ数

処理内容		A	B	C	合計ステップ
学習	学習エージェントの実装	14	3	0	17
	学習用のパラメータ設定	0	6	12	18
	合計	14	9	12	35
識別	識別エージェントの実装	14	3	0	17
	識別用のパラメータ設定	0	3	0	3
	合計	14	6	0	20

6. まとめと今後の予定

提案手法では，エージェントに対してディープラーニング機能を搭載し，ディープラーニングエージェントを開発するために有用であると思われる機能を実装した．今後は開発支援機構による開発に関してアンケート調査を行い，実装した開発支援のための機能が有用であるかを調査する．加えて必要な機能の追加，改善を行うことで，より優れた機構を構築する．

また現段階では，エージェントは予め用意された学習用データセットから学習を行い，得られた学習モデルによって識別を行った上で行動を決定している．そのため，エージェント自身が経験に基づいてよりよい行動を学習するという学習性を付与できておらず，学習用データセットを用意する必要性も生じる．そこで，今後はディープラーニングに強化学習である Q 学習を取り入れた Deep Q-Network[4] を導入することでエージェントが自律的に学習データを集め，行動選択を学習することを目指す．

参考文献

[1] 板津呂翔，打矢隆弘，内匠　逸，木下哲男，“知的マルチエージェントシステムにおける強化学習エージェント設計支援機構”，JAWS2012 講演論文集，2012.

[2] T.U. etal.，“Repository based multiagent framework for developing agent systems”，IGI Global，vol.Ch.4，pp.60–79，2011.

[3] Yann LeCun, Corinna Cortes, Christopher J.C. Burges, “Mnist handwritten digit database, yann lecun, corinna cortes, christopher j.c. burges”, http://yann.lecun.com/exdb/mnist/.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with Deep Reinforcement Learning”, NIPS Deep Learning Workshop 2013，2013.