Technology Reports

深層学習　コンテナ型仮想化　並列処理

短期間でAIモデルを作成・提供可能とする深層学習基盤の開発

サービスイノベーション部　伊吹（いぶき）　勇郎（たけろう）†

近年，深層学習を用いたAI開発がさかんに行われているが，高精度なAIを実現するために開発者は学習条件を変えながら試行錯誤し，複数回の学習処理を実行する必要がある．この試行錯誤が開発時間を長くする大きな要因となっている．また，深層学習を実行するためのフレームワークやライブラリも数多く存在し，それらの管理やバージョンアップも煩雑になっている．今回，ドコモではそれらの問題を解決し，短期間で高精度な学習を可能とするための深層学習基盤を開発した．本稿では基盤の詳細とその技術について解説する．

1. まえがき

画像認識や翻訳，音声認識などの処理を人工的にコンピュータに行わせる人工知能（AI：Artificial Intelligence）において，近年，ディープニューラルネットワーク（Deep Neural Network）を用いた深層学習技術が注目されている．ニューラルネットワーク[*1]自体は以前より存在していた技術であったが，精度に問題があり実用的ではなかった．しかし，GPGPU（General Purpose computing on Graphics Processing Units）[*2]などの計算機資源の大幅な進歩や，CUDA（Compute Unified Device Architecture）®[*3]，CuDNN（NVIDIA CUDA Deep Neural Network library）[*4]の登場によりニューラルネットワークの深層化に成功し，大幅な精度改善を達成するに至った．また，Tensorflow®[*5]やCaffe[*6]といったディープニューラルネットワークを開発するためのフレームワーク[*7]も数多く登場し，深層学習を用いたAI開発は多くの研究機関や企業で広く実施されるようになった．ドコモにおいてもdocomo Developer Supportで提供している画像認識APIなどのAIサービスで深層学習の技術を用いている．しかし，これらのAI開発において，高精度な結果を得るためには複数の条件で学習を実行する必要があり，加えて，それらが手動で行われているために，多くの人手が必要かつサーバの利用効率が悪いとい

©2018 NTT DOCOMO, INC.
本誌掲載記事の無断転載を禁じます．

† 現在，DOCOMO Innovations, Inc.

＊1 ニューラルネットワーク：人間の脳内にある神経細胞（ニューロン）とそのつながりを数式的なモデルで表現したもの．入力層，出力層，中間層から構成され，ニューロンや層の数，層の間のつながりの強さを変更することにより複雑な関数近似を行うことができる．

＊2 GPGPU：一般にコンピュータにおける画像の描画などの画像処理に用いられるGPUを画像処理以外の用途に転用する事．並列分散処理に優れる．

う問題があった．また，複数ライブラリ*8やフレームワークの環境構築・バージョンアップが煩雑になっているという問題も存在した．

そこでドコモでは，大量の条件を自動で並列処理することにより，短期間で高精度な学習を可能とし，また複数のフレームワークを容易に扱えるようにするために，コンテナ型仮想化技術*9と独自のスケジューリングシステムを用いた深層学習基盤を開発した．

本稿ではドコモが開発した深層学習基盤で用いられている技術の詳細と，本基盤によってもたらされる効果について解説する．

2. 深層学習とその課題

2.1 深層学習の概要

深層学習とは機械学習*10の一種であり，ニューラルネットワークの中間層を大量に用意したディープニューラルネットワークを用いた技術のことを指す．

深層学習は大量のデータからパターンやルールを習得する「学習」と，新たなデータに対して学習した結果からどのパターンに当てはまるかを推測する「推論」の2つの処理からなり，学習後のニューラルネットワークをAIモデルと呼ぶ．

AIモデルおよびサービスの事例として，画像データを用いた画像認識や，音声データを用いた音声認識や翻訳といったさまざまなものがある．

2.2 パラメータチューニングの課題

深層学習において，データさえ準備すればすべて自動で最良のパターンやルールを学習してくれるわけではなく，初期値や学習率といったハイパーパラメータと呼ばれる学習の条件を設定する必要がある［1］［2］．代表的なハイパーパラメータを**表1**に示す．

同じデータを用いてもハイパーパラメータが異なると学習が全く進まないことや，逆に特定のデータのみに偏った学習をしてしまい，汎用性が失われる過学習と呼ばれる問題が起きてしまうことがあり，最良のハイパーパラメータを選択するのは深層学習のモデル構築にとって非常に重要である．

しかし，ハイパーパラメータの種類は多岐にわたり，その試行錯誤を手動で実行するために多くの人的稼働がかかってしまうという課題が存在する．また，人手でハイパーパラメータを変更しながら学習

表1　代表的なハイパーパラメータ

ハイパーパラメータ	説　明
ミニバッチサイズ	1回の試行で処理するデータ数
学習率	重みを更新する幅
繰返し回数	試行数
重みの初期値	ニューロンの結合の強さの初期値
活性化関数	ニューロンの活性化方法
最適化関数	学習の進み方を決定する方法
ドロップアウト率	あるノードでニューロンを無効にする割合

＊3　CUDA：NVIDIAが提供するGPU向けの汎用並列プログラミング環境．NVIDIA Corp. の登録商標．
＊4　CuDNN：NVIDIAが公開しているDeep Learning用のCUDAライブラリ（＊8参照）．NVIDIA Corp. の登録商標．
＊5　Tensorflow®：深層学習プログラミング用のフレームワーク（＊7参照）．Google Inc. の登録商標．
＊6　Caffe：深層学習プログラミング用のフレームワーク（＊7参照）．
＊7　フレームワーク：ある領域のソフトウェアに必要とされる汎用的な機能や基本的な制御構造をまとめたもの．ライブラリでは，開発者が個別の機能を呼び出す形となるが，フレームワークでは，全体を制御するのはフレームワーク側のコードで，そこから開発者が個別に追加した機能を呼び出す形となる．
＊8　ライブラリ：汎用性の高い複数のプログラムを，再利用可能な形でひとまとまりにしたもの．

を実行しており，その際に手動で1台ずつサーバにログインして利用することで，サーバの利用状態把握を確認していることから，サーバ利用効率が悪くなり，結果を取得するのに多くの時間を要するという課題もある．

2.3 環境構築の課題

深層学習の，学習や推論処理は多数の行列演算で構成されている．そのため，演算装置としてCPUより大量のコアを搭載して並列処理が可能なGPUを用いた方が高速になることが知られており，NVIDIA社のGPUが広く用いられている．GPUはもともと画像描画処理に用いられていたが，高速演算のために深層学習やその他の演算でも用いられており，プログラミング環境やライブラリとして，汎用処理のためのCUDAや，深層学習のためのCuDNNが存在する．しかし，これらを扱うにはC言語を拡張した専用の言語を用いる必要があり，習得に時間がかかるため，大学や企業が深層学習プログラミングを扱いやすくするためのフレームワークを開発し，OSS（Open Source Software）[*11]として公開している．それらの代表的なものにTensorflowやCaffeといったものがある．これらのフレームワークは，データ分析で多く用いられているPython[*12]を用いて利用でき，ニューラルネットワーク構造を容易に記述するためのAPI（Application Programming Interface）[*13]が用意されているといった利点がある．

ただ，ライブラリとフレームワークは古いバージョンだと動作しない，処理速度が遅いといった問題が生じることがあるため，バージョンアップをする必要がある．しかし，それらを都度バージョンアップするのは手間がかかり，また，あるGPUでは特定のライブラリバージョンでしか動作しない，フレームワークごとに対応しているライブラリのバージョンが違う，といった依存性の問題が存在するため，容易にバージョンアップすることができない．

3. 深層学習基盤

3.1 課題解決に対するアプローチ

前述の2つの課題である，①ハイパーパラメータの選択が人力で非効率，②環境構築が煩雑，を解決するための手段と実際に開発した深層学習基盤のシステムについて以下に述べる．

⑴サーバ状態に応じたジョブの自動並列学習処理システムの開発

①の問題点として以下が挙げられる．

・パラメータ変更および学習の実行を手動で逐次行っていること

・各サーバの状態監視を手動で行っていること

これを解決するためにはジョブのキュー制御およびサーバ状態に応じたジョブの自動並列処理が有効であると考え，サーバを一元的に管理し，順次並列で学習処理を実行することが可能なシステムを開発することとした．

⑵Docker®[*14]を用いたコンテナ型仮想化技術の採用

②の問題点については以下が挙げられる．

・ライブラリやフレームワークのバージョンアップに時間がかかること

・ライブラリやフレームワークに依存関係があり，バージョンアップが容易ではない事

これらの解決手段として，サーバ内のジョブ単位で環境を独立に構築できる仮想化技術に着目した．今回は深層学習用のライブラリおよびフレームワークのみが必要であり，従来のホスト型仮想化技術[*15]のようなOS機能は不要であったため，コンテナ型仮想化技術，具体的にはDockerを用いることとした（**図1**）．DockerとはLinux®[*16]のコンテナ仮想化

＊9 コンテナ型仮想化技術：コンピュータ仮想化技術の一種で，1つのホストOSの上にコンテナと呼ばれる専用領域を作り，その中で必要なアプリケーションソフトを動かす方式のこと．

＊10 機械学習：サンプルデータから統計処理により，有用な判断基準をコンピュータに学習させる枠組み．

＊11 OSS：ソースコードが無償で公開されており，誰でも再利用や改変が行えるソフトウェア．ただし，原作者の著作権自体は放棄されておらず，派生物を作成した場合や再配布を行った場合にも，元の著作権表示を保持することが求められる．

＊12 Python：汎用のプログラミング言語の1つ．

＊13 API：あるプラットフォーム（OSやミドルウェア）向けのソフトウェアを開発する際に使用できる命令や関数を利用するためのプログラム上の手続きを定めた規約の集合．

＊14 Docker®：コンテナ型仮想化ソフトウェア．Docker Inc. の登録商標．

の技術を用いたソフトウェアで，OSが利用するリソースを隔離することで複数の環境を並立させることを可能にするものである．従来の仮想化技術であるハイパーバイザ型仮想化技術*17やホスト型仮想化技術に比べて，OSを新たに起動する必要がないため高速であり，一度作成したコンテナ用イメージは他環境に容易に移行可能というメリットも存在する．

3.2 システム概要

今回開発した深層学習基盤のシステム概要を以下に示す（**図2**）．

本基盤はMasterサーバとWorkerサーバ，File

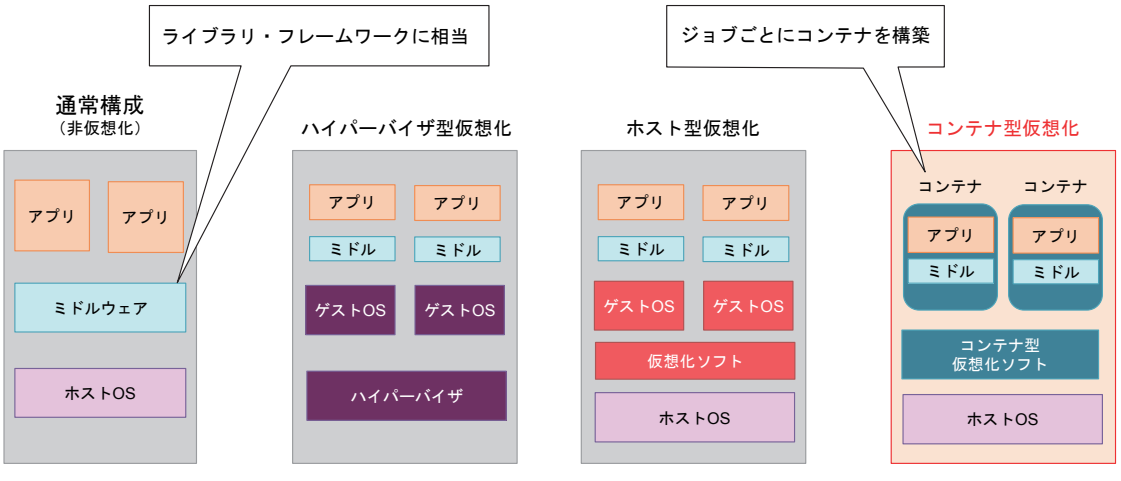

図1　コンテナ型仮想化技術と他仮想化技術の違い

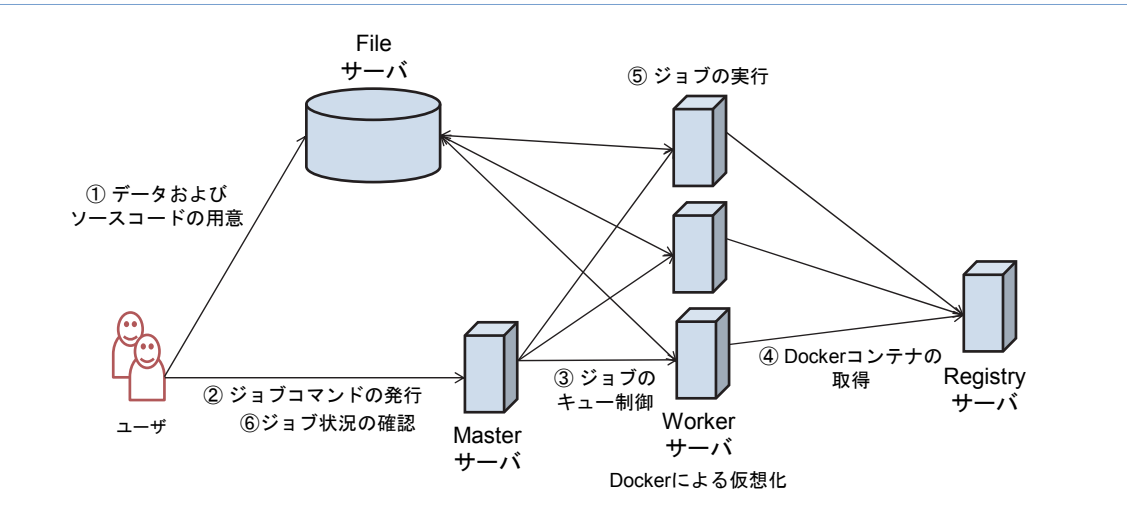

図2　深層学習基盤システム構成

*15 ホスト型仮想化技術：コンピュータ仮想化技術の一種で，サーバ上のOSに土台となるソフトウェアをインストールし，そのソフトウェア上で仮想マシンを稼働させる方式．

*16 Linux®：GPL（GNU Public License）に従って自由に再配布可能なUnix系のオープンソースOS．Linuxは，Linus Torvaldsの米国およびその他の国における登録商標あるいは商標．

*17 ハイパーバイザ型仮想化技術：コンピュータ仮想化技術の一種で，サーバへ直接インストールし仮想マシンを稼働させる方式．

サーバ，Registryサーバからなる．

・Masterサーバは深層学習ジョブの管理機能をもち，ユーザからの入力はすべてMasterサーバを経由して処理される．
・WorkerサーバはMasterサーバの命令を受けて学習を実行するサーバである．学習は各Workerサーバ内のDockerコンテナ上で実行される．なお，Docker単体ではWorkerサーバのGPUを利用する機能が含まれていなかったため，OSSであるNvidia-docker[*18]を利用している．
・Fileサーバは深層学習用のデータとソースコードを格納するためのサーバである．
・Registryサーバは，Dockerコンテナを一元的にに管理することができるDocker Registryを格納しておくためのサーバである．本基盤では多数の深層学習フレームワークおよびライブラリに応じたDockerコンテナを作成してRegistryサーバに格納することで，ユーザが任意のフレームワークおよびライブラリを選択できるようにした．

システム実行時にユーザが行う処理は深層学習用データとソースコードの格納，Masterサーバへのジョブの登録の2つであり，ジョブの制御および実行は自動で行われる．

3.3 ジョブ管理機能

ユーザから受け付けたジョブ管理をMasterサーバ上で行うために，今回以下の機能を開発した．

(a)ジョブ登録の受付（図2②）
(b)ジョブのキュー制御（図2③）
(c)ジョブ実行環境構築およびジョブ実行（図2③～⑤）
(d)ジョブ状態の把握（図2⑥）

まず，ジョブ登録とジョブのキュー制御機能を実現する方法として，メッセージキュー[*19]を使う方法と，RDB（Relational DataBase）[*20]を使う方法の2種類を検討した．メッセージキューの方がRDBと比較して高スループット・低遅延というメリットが存在する一方，ジョブの状態監視やリトライ処理が煩雑になるというデメリットがあることが分かった．今回はジョブの状態監視が重要，かつ実際のジョブを実行するのはWorkerサーバでありジョブ登録およびキュー制御にスループットや遅延はさほど重要ではないため，RDBを利用する方法を採用した．

次に，具体的にそれらの機能をどう実現しているかを解説する．

ユーザは深層学習処理に必要なデータとソースコードを用意（図2①）する．その後Masterサーバに対してあらかじめ定められた形式のジョブコマンドを発行する（図2②）と，Masterサーバ内に構築したRDBにレコードがキューとして追加される．ジョブコマンドおよびオプションは本基盤用に独自に開発したものであり，深層学習フレームワークの選択やFileサーバ上のデータやソースコードの場所を指定することができる．

次にジョブのキュー制御だが，GPUメモリの制限の関係から1つのGPU上で実行するジョブの数は1つであることが好ましい．よって定期的にMasterサーバが各WorkerサーバのGPUの利用状態を確認し，ジョブが実行されていない場合キューの先頭にあるジョブを順次実行する（図2③）という制御を実装した．キュー制御によって実行可能と判断されたジョブについて，該当のWorkerサーバは指定されたDockerコンテナをRegistryサーバから取得（図2④）し，ジョブを実行する（図2⑤）．

最後にジョブ状態の把握であるが，こちらはRDBのテーブルをユーザが参照することで把握することとした（図2⑥）．なお，前述の通り(b)，およ

＊18 Nvidia-docker：dockerの拡張ソフトウェア．
＊19 メッセージキュー：アプリケーションソフト間でデータを交換して連携動作させる際に，送信するデータをいったん保管しておき，相手の処理の完了を待つことなく次の処理を行う方式．
＊20 RDB：関係データベースと訳され，データを複数の表として管理し，表と表の関係を定義することで，複雑なデータの関連性を扱えるようにしたデータベース管理方式．

び(c)に関しては自動的に制御・実行される．

4. 効　果

本基盤の導入により，達成された効果は以下3点である．

(1)AIモデルの学習時間短縮

これまでのAIモデルの開発は，人手での実行かつ学習条件ごとに各サーバでまとめて実行していたため，**図3**(a)のようにサーバの利用効率に大きな無駄が発生していたが，本基盤でジョブのスケジューリングを自動化し，並列処理ができるようになったことにより，学習時間の大幅な短縮が実現された(図3(b))．

(2)学習精度の向上

AIモデルの学習効率が高まり，数多くの条件で学習を試すことができるようになったことにより，学習精度の大幅な向上を達成した（**図4**）．

(3)環境構築の短縮化

Dockerを利用することにより，新しい深層学習フレームワークの導入やバージョンアップが容易になった．具体的にはこれまで数時間かかっていた環境構築が1／10に短縮された．

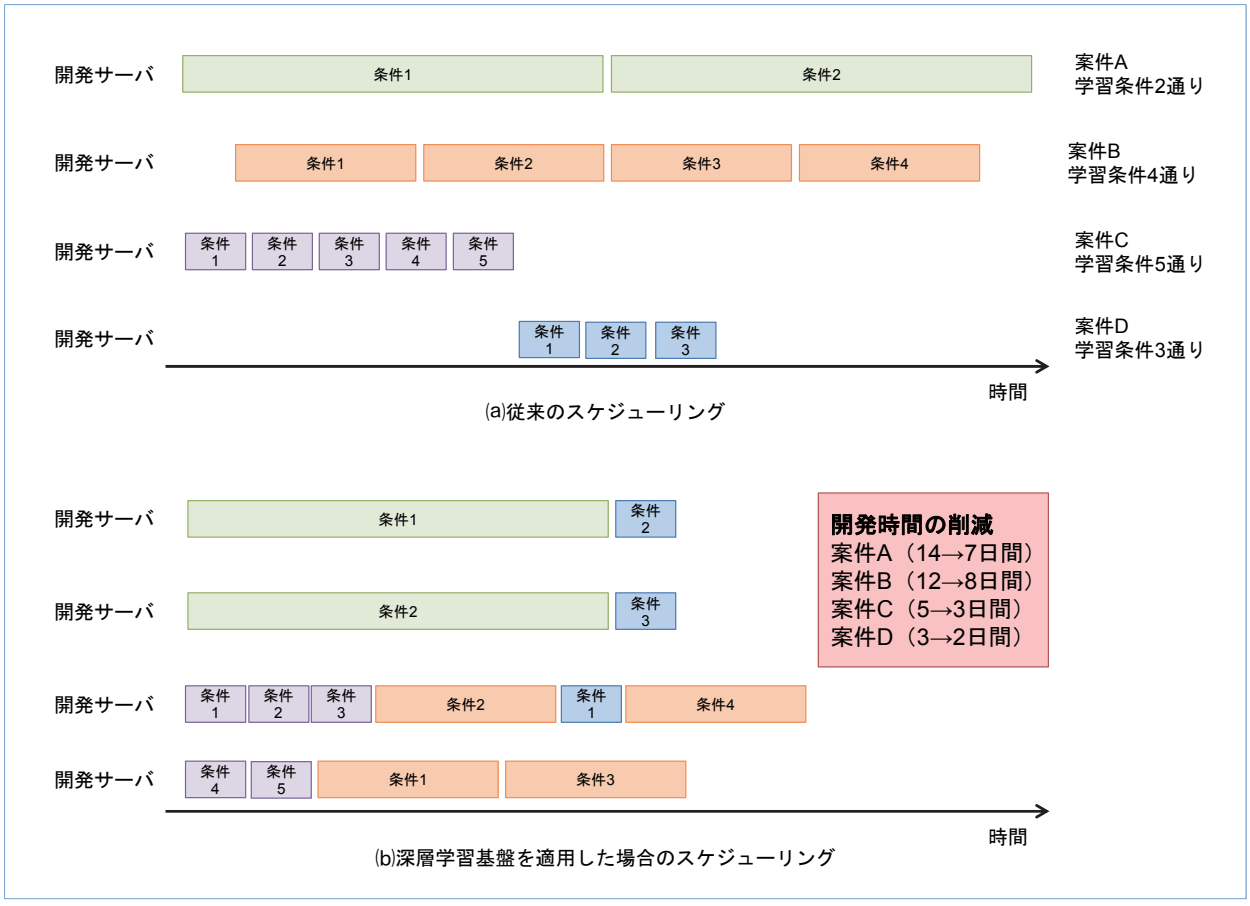

図3　深層学習基盤を適用した場合の効果

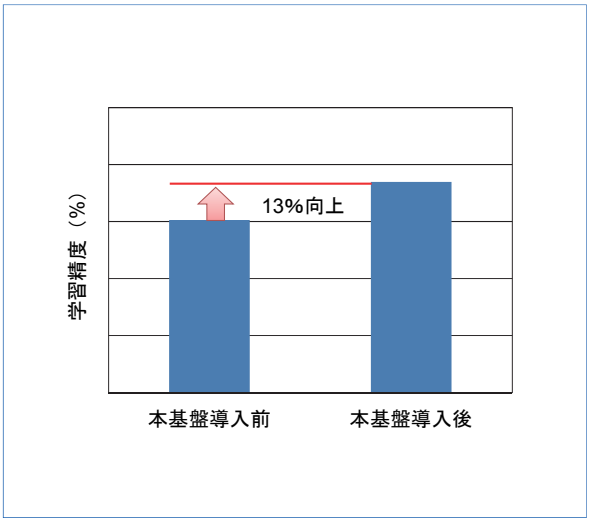

図4　深層学習基盤を利用することによる精度向上

5. 今後の展望

今後の展望としては以下の3点が挙げられる．

⑴複数世代GPU・複数クラウド環境への対応

GPUの世代に関してはほぼ毎年進化しており，それに応じて扱えるモデルの大きさが変わってくるためGPUの世代を考慮してWorkerサーバへのジョブ分配を実施しなければならない．また，複数のクラウド環境でも扱えるように整備を行う．

⑵スケジューラの強化

現在のジョブスケジューラでは単純な優先順位付けしかできないが，複雑な優先順位付けを可能とするように改良を行う．

⑶パラメータ最適化

現在の学習は複数条件を並列処理させるだけであるが，この場合条件を多くすればするほどサーバを専有してしまう．計算量を削減するために効率よく最適なパラメータの組合せを決定できるようなアルゴリズムの技術確立を行い，より高精度なAIモデルを短時間で開発可能とする。

6. あとがき

本稿では，短期間で高精度なAIモデルを開発するための深層学習処理を可能とする基盤について，技術的詳細を解説した．本基盤を用いることにより深層学習への取組みを加速させると同時に，基盤自体も進化させ，さらなる効率化を図る．

文　献

［1］　岡谷 貴之：“深層学習，”講談社，2015.

［2］　Yoshua Bengio: “Practical recommendations for gradient-based training of deep architectures,” Neural Networks, Springer, pp.437–478, 2012.