解説

ディープラーニングとGPUのチュートリアル

Tutorial for Deep Learning and GPU

森 野 慎 也* *エヌビディアジャパン

Shinya Morino* *NVIDIA Japan

1. ディープラーニング

2014年のILSVRC (IMAGENET Large Scale Visual Recognition Challenge) にて、画像からの一般物体認識の精度は、人間と比肩する精度に達した。これを実現したディープラーニングが大きな話題となり、昨今のAIブームの引き金となった[1]。その用途は、クラウド上のサービスから、自動運転にまで広がっている。合わせて、ディープラーニングのサイエンスや技法も急速に発展している。

ディープラーニングとは、ニューラルネットワークの発展形である。ニューラルネットワークの一般的な例として3層パーセプトロンが(図1(a))があり、隠れ層と呼ばれる層を持つ。そして、多数の隠れ層を持つニューラルネットワークを、ディープニューラルネットワークと呼ぶ(図1(b))。

ディープラーニングと呼ばれる分野では、このディープニューラルネットワークが使用され、その動作は、ネットワーク形状、および、与えられる係数の値により決定される。操作は、すべて数値演算により行われる。

ディープニューラルネットワークでも、通常のニューラルネットワークと同様に、学習(トレーニング)、推論(インファレンス)という二つの段階を持つ。学習では、ニューラルネットワークの形状を決定し、学習させたい対象を含む大量の学習データを用いて、係数を決定する。推論では、ネットワークの入力に評価対象となる入力を与え、学習時に決定された係数を用いて評価を行う。

ディープニューラルネットワークは、通常のニューラルネットワークと大きく扱いは変わらないものの、層数が多いため、学習が難しいといわれてきた。大きな進展が見られたのは、ここ数年のことである。

冒頭でも触れたが、2012年に、IMAGENETにおける一般物体認識コンテスト(ILSVRC)において、ディープラーニングが大きく認識精度を向上させた[2]。ここで用いられ

図1 ニューラルネットワークの構造

たネットワークは、AlexNet と呼ばれ、今でも多く引用されるディープニューラルネットワークである[3]。続いて現れたVGG[4]やGoogLeNet[5]、ResNet[6]など、現在ではリファレンスとも呼べるネットワークが現れ、さらに、認識精度を向上させた。

これまで、画像認識、音声認識の分野では、人間が特徴量を設計し、機械学習の手法を組み合わせることで、認識機能を実現してきた。一方、ディープラーニングでは、大量のデータを用いた学習過程で、特徴量がディープニューラルネットワーク上に獲得される。人間では網羅的に分析不可能な、大量のデータを使用して、数値処理を行い、非常に高い水準の認識精度を実現しているのである。

ディープラーニングは、画像であれば、物体認識のみならず、物体検知(Object detection)[7][8]、画像のセグメンテーション[9]、自然言語処理においては、音声認識[10]~[12]、翻訳[13]、Web上のサービスとしては、リコメンデーションエンジン[14]、ほかにも、医療・創薬を含め、多くの分野で有用であることが知られている。また、文書から画像を生成する[15]、動画から対応する音を生成する[16]

原稿受付 2017年2月3日

キーワード: Deep Learning, GPU, Development, Tutorial

〒107-0052 港区赤坂2-11-7

*Minato-ku, Tokyo

など、データの生成機能を持たせることもできる。現時点でも、新規の手法、新規の応用が生まれ続けている。

大きな広がりを持つディープラーニングであるが、本稿では、GPU という観点から、ディープラーニングにおける開発の流れ、および、その特性について、述べてゆく。

2. ディープラーニングにおける GPU の活用

GPU (Graphics Processing Unit) は、ディープラーニングにおけるキーデバイスとなっている。GPU は、もともと、グラフィックスの描画に特化した専用プロセッサであった。しかし、NVIDIA から 2006 年の CUDA (Compute Unified Device Architecture) [17] が発表され、非常に高い並列性を持つ汎用演算が、プログラミングにより実行可能となった。すでに、スーパーコンピュータの分野では、広く採用されている。

2.1 ディープラーニングにおける GPU の活用

ディープラーニングにおける GPU の有用性が知られるようになった、先駆的な事例として、「Google の猫」が挙げられる [18]。この報文では、オートエンコーダと呼ばれる、入力と出力が同値となるディープニューラルネットワークを利用している。オートエンコーダは、ニューラルネットワークを用いた次元削減の方法として知られており、入力の次元数より少ない次元で、与えられた学習データの特徴を抽出し、表現することができる。「Google の猫」の例では、学習されたネットワーク中に、猫や人の顔に対応するニューロンが生成されていることが分かった。これから、画像を再構成したものが、図 2 である。これは、先験的な知識なしに、猫の顔を抽出できたことを示している。

学習には、1,000 台のコンピュータ (16,000 個の CPU コア) が用いられた。しかし、A. Coates らにより、16 ノード、64 台の GPU (4 GPU/ノード) でも、同等の処理が実現できることが示された [19]。GPU を活用することで、Google のような大きなデータセンターを持たなくとも、ディープ

図 2 再構成された猫のイメージ

ラーニングに対する研究・開発を行えることが示され、大きな注目を集めた。

2.2 ディープラーニングフレームワーク

今日では、多数のディープラーニングフレームワークがオープンソースとして配布されており、そのほとんどが、GPU を活用している。

ディープラーニングフレームワークの先駆者ともいえる Caffe [20] では、初期から GPU に対応した実装が行われている。初期の Caffe の実装では、GPU (CUDA) で実行される BLAS ライブラリ (cuBLAS)、および、乱数生成ライブラリ (cuRAND) が使用されていた。加えて、今日の Caffe では、NVIDIA により提供される cuDNN ライブラリにより、他のレイヤーに対して高速化された実装が提供されている。さらに、NVIDIA からは、オリジナルの Caffe の fork として、NVCaffe が提供されている [21]。ほかに、多くのフレームワークがリリースされている。NVIDIA のホームページ [22] をご参照いただきたい。

3. ディープラーニングにおける開発とデプロイ

本章では、ディープラーニングの学習における、ワークフローを確認してゆく。具体的な例としては、画像認識を題材とし、手書き文字データセットである MNIST [23] を用いる。LeCun らによる、LeNet と呼ばれるコンパクトなディープニューラルネットワークにより、よく認識できることが知られている [24]。

NVIDIA からは、NVIDIA DIGITS [25] と呼ばれる、Web GUI を備えたディープラーニングの開発環境が提供されており、MNIST を使ったトレーニングのチュートリアル [26] が提供されている。ぜひ、試していただきたい。

3.1 学習の準備

ディープニューラルネットワークを用いた開発では、以下の準備が必要となる。

- (1) 学習用データの準備
 - (2) ディープニューラルネットワークの選定
 - (3) 学習用ハードウェア (PC) の準備
 - (4) ディープラーニングフレームワークの選定
- ・学習用データの準備、アノテーション

まず、学習用のデータを作成する。画像の場合、サイズやフォーマットを合わせることも含め、前処理を行う。また、データに正解ラベルを与える必要があり、これをアノテーションと呼ぶ。今回のデータは、MNIST であり、 28×28 ピクセルのグレースケール画像である (図 3)。すでに、画像と正解ラベルとしての数値の対応づけも行われている。

- ・ディープラーニングフレームワークの選定

ディープラーニングにおける学習を行う際には、ディープラーニングフレームワークと呼ばれるソフトウェアが用いられる。今回は、Caffe を用いる。Caffe は、非常にポピュ

ラーなフレームワークのうちの一つである。これまでの報文においても、報文の成果を再現するための設定やパッチなどが、インターネット上で入手可能である。加えて、よくチューニングされており、実行も高速である。画像系のディープラーニングにこれから取り組む研究者、技術者の方々には、取り急ぎ、Caffe をお勧めしたい。

・ディープニューラルネットワークの構造決定

自らの目的に適したディープニューラルネットワークを選ぶ（もしくは、作り上げる）。ネットワークを選ぶためには、よく知られているネットワーク（AlexNet, GoogLeNet, VGG など）を転用する、類例の論文・発表から、構造を類推するなどの作業を行う。arXiv など、オープンな論文サイトからも、多くの参考になる報文を入手することができる。

今回用いるディープニューラルネットワークを、図 4 に示す。これは、DIGITS で事前に登録されているネットワークであり、Y. LeCun らによる報文に記載された構造から派生している [24]。

まず、ネットワークの入力段では、 28×28 ピクセルの画像に、20 とおりのコンボリユーション（サイズ 5×5 ）を適用することで、20 枚の特徴マップ（feature map）を生成する。この後、 2×2 ピクセルの領域から最大値を取得し、得られた特徴マップのサイズを、半分（ 12×12 ピクセル）にする（最大値プーリング）。次段のコンボリユーション、最大値プーリングへとつないでゆく。最終的に、特徴マップは、50 枚の 4×4 ピクセルのイメージとなり、複数の全結合層への入力となっていく。最終的には、画像に対応する 0~9 という認識結果が得られる。ネットワーク中に現れるコンボリユーション層、全結合層における係数は、学習

により決定される。

・学習用ハードウェア（PC）の準備

学習に用いる PC は、x86 系の CPU を積んでいる通常の PC でよい。ただし、学習の演算を加速するために、GPU を搭載している必要がある。また、多くのディープラーニングフレームワークは、Ubuntu Linux 上で動作する（Linux になれていない方も多数いると思われるが、ぜひ、チャレンジしていただきたい）。

現在、NVIDIA がディープラーニング向けにサポートしているのは、Kepler と呼ばれる世代以降の GPU である。確認するには、“CUDA GPUs” ページ [27] から、GPU の Compute Capability と呼ばれる GPU の世代を示す番号を確認する。Kepler GPU であれば、メジャー番号が 3、つまり、3.x と表される。つまり、Compute Capability が 3.0 以降の GPU を選択いただければよい。新規に GPU を入手される方には、Pascal 世代の GPU（Compute Capability 6.x）をお勧めしておく。

3.2 学習

学習の過程で、ディープニューラルネットワーク中の係数が決定され、妥当な推論を行うことができるようになる。ここで達成すべき要件は、精度である。文字認識であれば、認識精度が満たすべき要件となる。

そもそも、現実世界の手書き文字に対して、数学的に厳密な認識基準を定義することはできない。ディープニューラルネットワークは、手書き文字という入力に対して、0~9 という数字を対応づける関数とみなすことができるが、ここでは、十分に精度のよい、近似を得ることが目的となる。

学習は、学習データの正解と、ネットワークによる算出結果が同じになるように行われる。学習データの正解とネットワークによる算出結果の差を、一般的に less 関数と呼ぶ。言い換えれば、less 関数の値が最小となるよう、ネットワーク中の係数を更新していくのが、ディープラーニングにおける学習となる。演算の流れとしては、まずは、順方向に演算を行い、推論結果を得る。もし、推論結果が間違っている場合には、less 関数により得られる誤差を、出力側の層から逆に伝搬してゆき、各層の係数を更新する。この演算は、誤差逆伝搬法（バックプロパゲーション）と呼ばれる。

学習の際には、学習データと検証データの二つを用いて行われる。学習データは、上記の係数を決定するための演

図 3 MNIST に含まれる手書き文字例

図 4 今回使用するディープニューラルネットワーク

図5 ディープニューラルネットにおける演算

算に用いられる。検証データは、学習が一般性を失わずに進行していることを確認するために用いる。図5では、学習時の loss 関数と精度のグラフの例を示す。学習データからの loss 関数の値が減少するとともに、検証データの loss 関数の値も同様に減少する。これが、一般性を失わず進行している学習の例となる。また、検証データを用いた精度 (Accuracy) の算出を行い、正答率も合わせて確認する。

精度を向上させるために、ハイパーパラメータの変更を行いつつ、学習を繰り返し試行する。ハイパーパラメータのうち、代表的なもののうちの 하나가学習率であり、これにより、演算の収束速度を意図的に遅らせる。ほかに、バッチ数は、学習データを指定された数だけ取り出し、演算を行う (ミニバッチ法)。これらのパラメータは、学習が局所解に収束し、認識精度が得られなくなる事態を防ぐために、用いられる。試行の後にも、認識精度が得られない場合には、ネットワークの構造そのものも、変更して、試行を行う。

3.3 推論用の実装作成

学習が完了したディープニューラルネットワークは、アプリケーションやサービスに展開される。推論時の要件は、処理性能である。つまり、必要とされる処理レイテンシ、もしくは、処理スループットを満たしていることが、ゴールとなる。

推論は、図5に示したように、ディープニューラルネットワークの順方向の演算であり、学習中にもフレームワークにより行われている。つまり、推論演算を行う実装として、フレームワークの順方向の演算をそのまま転用することができる。もし、性能要件が満たせれば、このままデプロイすることも可能である。

ただし、フレームワークの推論における実行性能は、一般にあまり高くない。これは、多種のディープニューラルネットワークを汎用的に扱うための実装から生じる、オーバーヘッドによるものである。しかし、学習完了時点で、ネットワーク構造は決定されており、この構造に特化し、オー

図6 学習曲線

図7 開発における繰り返し

バーヘッドを取り除くことが可能である。加えて、コンボリューションサイズなど、ネットワークのパラメータに応じて、GPU 上で性能が出る実装を選択することで、さらに性能を向上させることができる。

この目的のために、NVIDIA からは、TensorRT という製品 [28] が発表されており、ディープニューラルネットワークの実行性能最適化を、実行する GPU に合わせて行うことができる。入力として、Caffe のネットワーク定義ファイルを与えることで、自動的に推論エンジンのチューニングを行うことが可能である。

また、最適化を行っても、実行性能を満たせない場合がある。この場合には、学習に戻り、ネットワークの形状を「許容できる範囲で」簡略化するなどとして、演算量を減らしてゆく。この意味では、学習と推論エンジンの作成も、繰り返しのプロセスとなる場合がある。

3.4 アプリケーション・サービスへのインテグレーション

GPU には、CUDA で使用可能な汎用演算のみならず、動画のエンコード・デコードをはじめとして、種々の機能が搭載されている。これらの機能を活用することで、本来の GPU の性能を最大限に活かすことが可能となる。ここでは、推論エンジンのアプリケーション・サービスへのインテグレーション時に、利用できる技術について、簡単に説明する。

・ GPUDirect For Video [29]

PC の場合、GPU は、PCI express バスを介して CPU

に接続されており、CPU 側のメモリと GPU 上のメモリは、物理的に異なる。推論の対象となるデータは、PC 外部から取得されるが、効率よく、GPU へと転送する必要がある。このために、GPUDirect For Video を用いることができる。GPUDirect For Video では、CPU 側でデータを受信したメモリから、直接、GPU へとデータを DMA 転送することができる。これにより、処理レイテンシを減らすことが可能となる。

・動画エンコード・デコード

GPU 上には、動画のハードウェアエンコーダ、デコーダが搭載されている [30]。

PC では、動画を圧縮されたままの状態 GPU にデータを送り込み、GPU 上でデコードすることで、後続する処理を効率よく行うことができる。また、エンコードも GPU 上で実施し、圧縮されたデータを CPU 側で取得することも可能である。また、組み込みプラットフォームである Jetson では、同等の機能を GStreamer 経由で利用することができる [31]。

監視カメラをはじめとする、動画処理において、有効な手法となり、NVIDIA Deep Stream SDK [32] においても活用されている。

4. 最 後 に

大きく、ディープラーニングと、その開発過程について概観してきた。一点、注記すべきは、ディープラーニングにおける開発サイクルと、一般のソフトウェア開発における違いである。

ディープラーニングでは、学習の繰り返しにより、精度を改善してゆくことが、開発プロセスに対応し、データとディープニューラルネットワークが主役となる。一方、通常のソフトウェア開発は、仕様、実装、検証の段階を踏んで、明示された処理手続きを積み上げることで進められていく。

両者の間には、大きな考え方、進め方の違いがあり、ソフトウェア開発に慣れた方には、ディープラーニングを含めた開発計画の立案を考えた際に、困惑される方が多いようである。筆者もこの点について、よく質問をいただく。ただし、両者が、それぞれ、異なるパラダイム上に立脚していることを頭に置けば、ご理解いただけるものと思う。

また、ディープラーニングでは、大量のデータを用いた学習における過程で、特徴量がディープニューラルネットワーク上に獲得される。データ量に関しては、特徴量設計などの経験が必要な作業については、人間が個別に対応することは、もはや、現実的ではない。言い換えれば、ビッグデータの時代には、ディープラーニングのような、人間を介さず、コンピュータによる演算を活用し、AI としての性能を獲得できるディープラーニングに、アドバンテージ

があるといえる。データが大量になることで、演算量が劇的に増えるが、GPU を用いることで、処理が加速される。

本日、紹介させていただいたのは、ほんの触りだけである。1 年経てば古いのが、今日のディープラーニングである。逆に言えば、いつでも遅いことはない。本稿をご一読いただいた読者の皆様も、試してみられることを願う。

参 考 文 献

- [1] IMAGENET Large Scale Visual Recognition Challenge (ILSVRC) 2014, Introduction, http://image-net.org/challenges/LSVRC/2014/slides/ILSVRC2014_09_12_14_det.pdf
- [2] A. Krizhevsky, I. Sutskever, G. Hinton: <http://image-net.org/challenges/LSVRC/2012/supervision.pdf>
- [3] A. Krizhevsky, I. Sutskever, G. Hinton: <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- [4] K. Simonyan, A. Zisserman: ICLR 2015, <https://arxiv.org/pdf/1409.1556.pdf>
- [5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich: <https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf>
- [6] K. He, X. Zhang, S. Ren, J. Sun: <https://arxiv.org/pdf/1512.03385v1.pdf>
- [7] S. Ren, K. He, R. Girshick, J. Sun: <https://arxiv.org/abs/1506.01497>
- [8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, A. C. Berg: <https://arxiv.org/abs/1512.02325>
- [9] E. Shelhamer, J. Long, T. Darrell: <https://arxiv.org/abs/1605.06211>
- [10] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Y. Ng: <https://arxiv.org/abs/1412.5567>
- [11] D. Amodei, R. Anubhai, E. Battenberg, et al.: <https://arxiv.org/abs/1512.02595>
- [12] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, G. Zweig: <https://arxiv.org/abs/1609.03528>
- [13] Y. Wu, M. Schuster, Z. Chen, et al.: <https://arxiv.org/abs/1609.08144>
- [14] Apache Spark と Amazon DSSTNE を使った、Amazon 規模のレコメンデーション生成, <http://aws.typepad.com/sajp/2016/07/generating-recommendations-at-amazon-scale-with-apache-spark-and-amazon-dsstne.html>
- [15] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, D. Metaxas: <https://arxiv.org/abs/1612.03242>
- [16] Artificial intelligence produces realistic sounds that fool humans, MIT News, <http://news.mit.edu/2016/artificial-intelligence-produces-realistic-sounds-0613>
- [17] NVIDIA CUDA Zone: <https://developer.nvidia.com/cuda-zone>
- [18] Using large-scale brain simulations for machine learning and A.I., Google Official Blog, <https://googleblog.blogspot.jp/2012/06/using-large-scale-brain-simulations-for.html>
- [19] A. Coates, B. Huval, T. Wang, D.J. Wu, A.Y. Ng: http://ai.stanford.edu/~acoates/papers/CoatesHuvalWangWuNgCatanzaro_icml2013.pdf
- [20] Caffe, <http://caffe.berkeleyvision.org/>
- [21] NVCaffe, <https://github.com/NVIDIA/caffe>
- [22] NVIDIA Accelerated Computing, Deep Learning Framework, <https://developer.nvidia.com/deep-learning-frameworks>
- [23] The MNIST database of handwritten digits, <http://yann.lecun.com/exdb/mnist/>
- [24] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner: <http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf>
- [25] NVIDIA DIGITS, <https://developer.nvidia.com/digits>

- [26] NVIDIA DIGITS, Getting Started, <https://github.com/NVIDIA/DIGITS/blob/master/docs/GettingStarted.md>
- [27] CUDA GPUs, <https://developer.nvidia.com/cuda-gpus>
- [28] NVIDIA TensorRT, <https://developer.nvidia.com/tensorrt>
- [29] NVIDIA GPUDirect For Video, <https://developer.nvidia.com/gpudirectforvideo>
- [30] NVIDIA Video Codec SDK, <https://developer.nvidia.com/nvidia-video-codec-sdk>
- [31] Tegra X1/Tegra Linux Driver Package Multimedia User Guide —Nvidia, http://developer.download.nvidia.com/embedded/L4T/r23_Release_v1.0/L4T_Tegra_X1_Multimedia_User_Guide.pdf
- [32] NVIDIA DeepStream SDK, <https://developer.nvidia.com/deepstream-sdk>

森野慎也 (Shinya Morino)

2001 年より地震に関する計測機、システムの開発に従事。2013 年エヌビディア入社。GPU 上の並列演算環境である CUDA の普及、テクニカルサポートに従事する。2016 年よりディープラーニング部に所属。ディープラーニングを含めた GPU のテクニカルサポート

を行う。