AI画像認識のエッジ化とその実装技術

画像認識エンジンをエッジ・デバイスで実行することのメリットは、高速なレスポンスが期待できることで、特に製造現場における目視検査などで求められています。このようなデバイスを実装するには、ディープ・ラーニング技術を用い、大量の画像を学習して得られる学習済みモデルを準備しなければなりません。そこで、学習するためのシステムに求められる要件を整理し、実装を容易にするパッケージ・ツールの紹介を交えながら、エッジ・デバイスにデプロイして稼働させるまでの全体像の詳細を解説します。

1. AI画像認識の全体像

昨今のAIブームにおいて、最も注目され実用化が進んでいる技術分野の一つに画像認識のAI化があります。その中でも、「製造現場の目視検査にAI画像認識を使えないか」というお問い合わせは、筆者がAI技術に携わるようになってから最も多いケースです。例えば、「製造業の現場では機械化・自動化が進んでいるが、検査工程は人手で行っている」といった場合に、AI画像認識で判定をサポートしたいというものです。

このとき、「AIという不思議な力によって人智を超えた判定ができるのではないか」という過度の期待が伴っていることがよくあります。しかし、現場の検査員の方々の眼力はそう簡単に超えられません。実際に要求されることは、ごくわずかな傷や色ムラ、形状の歪み、光の当て方によってピンポイントでしか見つけることができない表面の欠陥の判定であることも多いのです。AI画像認識そのものの技術的難易度が高いため、現場ごとのノウハウを総合的に理解し、AI技術の採用に向けて今できることから始めよう、というのが今日の状況だと思います。

では、まずAI画像認識の全体像をざっと俯瞰してみましょう（**図1**）。ここでは、AI画像認識の主流であるディープ・ラーニングを用いています。ディープ・ラーニングの詳細については割愛しますが、最も“人工知能（AI）”らしい手法で、ヒトの脳神経回路に着想を得たアルゴリズムです。われわれが「今見えているものは何か」を認識する過程をイメージしてみてください。網膜に写っているものを基に、それを脳神経回路で処理して、これまでに見た記憶のある類似したものに帰着させることによって、何であるかを判別しています。これをコンピューターで再現するのがAI画像認識です。ディープ・ラーニングのアルゴリズムにたくさんの画像データを投入することによって「学習」し、その結果得られた「学習済みモデル」を実行して、別の画像データについて「推論（判定）」する、という流れになります。つまり、ヒトが今までに見たことのないものを何であるか認識できないように、AI画像認識においても、事前に画像データをたくさん与えて学習することが必須です。

具体的な例として、製造現場での目視検査でAI画像認識を用いて不良品検出をサポートするユースケースを考えてみましょう（**図1①**）。まずは、学習のための画像データをたくさん集めなければなりませんが（**図1②**）、ここに大きな障壁があります。不良品を検出する画像認識エンジンを作る場合、その不良品が写った画像が必要になりますが、不良品の画像など残していないというケースもあります。これから撮影しようにも、現場に撮影する仕組みそのものがないケースもあります。繰り返しになりますが、ヒトが今までに見たことのあるものと比較

することでしか認識できないのと同じようにAI画像認識も振る舞うため、現場で養った目、記憶を転用して、画像を集めることから地道に始めるしかありません。

さて、画像が集まったとします。次にその画像に写っているのは何なのかを文字情報として与えます。これをラベル付け、タグ付けもしくはアノテーションと言い、例えば不良品が写っている画像を「不良品」フォルダーに集めたり、欠陥が写っている箇所に座標情報として枠で囲むような作業をします（**図1③**）。実はこの工程も繰り返し続けなければならない、極めて地道な作業です。

ここまで来ると、ようやくディープ・ラーニングによる学習を行うことができます（**図1④**）。しかし実際には、Pythonをはじめとしたプログラミング技術を基にして、ディープ・ラーニングが実行できるライブラリー群であるフレームワークを呼び出すようなカスタム開発が必要です。ある程度はインターネットや書籍の情報を基にして開発できますが、通常はプログラミングのスキルやフレームワークAPIの知識があるエンジニアがいないと難しいでしょう。

ちなみにこの学習ジョブは大きな計算能力を要するバッチ処理で、GPUによる補助プロセッサーで高速化し、長時間の負荷に耐えられるようなサーバーで実行することが一般的となっています。また、その結果として得られる学習済みモデルの認識精度は、学習率やバッチサイズ、学習イテレーション数といったハイパー・パラメーターと呼ばれるチューニングによって影響を受けます。もちろん、入力する画像データ群とそのラベル情報の品質にも依存します。つまり1回の試行で最良のモデルが得られるとは限らず、さまざまな試行錯誤によってさらに大きな計算能力が要求されることもあります。

ここまで来て、ようやく1つの学習済みモデルを作ることができます。学習時のようなPythonなどのプログラミングによって、テスト画像を引数に与えて実行すれば認識結果が文字情報で返されます（**図1⑤**）。これは機械学習による推論の一つなので、確信度（スコア）が含まれます。95%の確率で不良品である、(50,100) と (100,200) の座標で囲まれた位置に98%の確率で凹み不良がある、といった判定です。そしてその後、実際の業務へ組込み、AIエンジンとして運用が可能になります（**図1⑥**）。

しかし、これで終わりではありません。誤認識してしまう画像を正しく認識できるようにモデルをアップデートしたいといった改善要求や、新製品が認識できるようにしたい、新たな不良モードが見つかった、といった新たなニーズへの対応要求もあるでしょう（**図1⑦**）。こういった改善のサイクルは引き続き必要で、画像データと学習済みモデルの世代管理のような運用設計も見越した持続的なITシステムが求められています。

2. AI画像認識のエッジ化

本章では、AI画像認識のエッジ（Edge）化について述べます。エッジ化によるメリットは、現場で得られた画像データを外部の環境に転送することなく、その場の認識エンジンで判定を行えるため、高速なレスポンスが

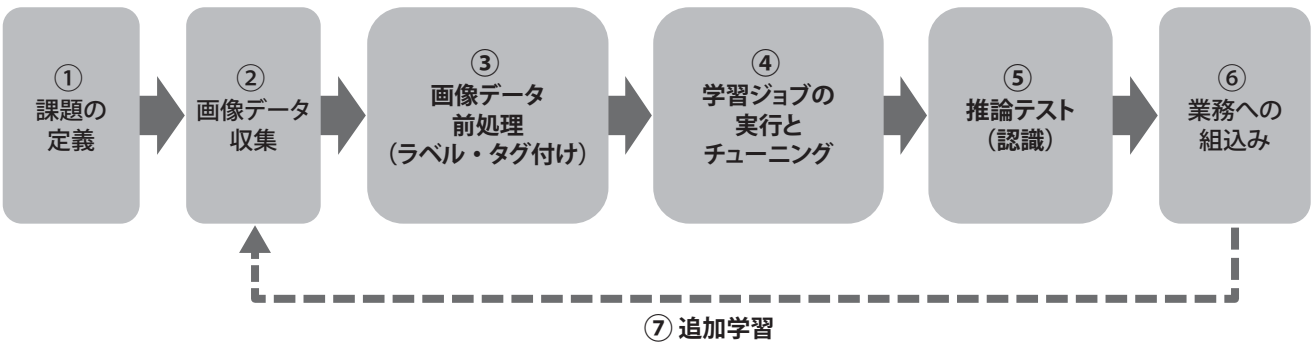

図1. AI画像認識のワークフロー

期待できることです。ここで認識エンジンを構成するものは、ディープ・ラーニング・フレームワークを実装した認識用のプログラムと、認識したい物体を記憶・経験した学習済みモデルです。この学習済みモデルはディープ・ラーニングのフレームワークによって形式が異なるものの、フレームワークが同じであれば環境の移植性があります。例えばTensorflowの学習モデルであるcheckpoint形式のデータは、同じTensorflowが動作する環境（NVIDIA Jetsonなどのエッジ用のボードやスマートフォン）にコピーして実行することができます。スマートフォンを例に挙げると、**図2**はiPhone上のTensorflow実装をアプリとしてパッケージしたものを実行している様子で、iPhoneのカメラでキャプチャーした画像から、アプリ内に格納した学習済みモデルを実行することにより、認識結果を重ねてリアルタイムに表示しています。外部システムに画像データをネットワーク送信して判定させているのではありません。ここではオフライン・モード（フライト・モード）で実行しています。ただし、認識のアルゴリズムは学習よりもはるかに軽量の処理ですが、専用のGPUを内蔵していないシステム環境ではあまり高速に動作しませんので注意が必要です。

現場で求められる形態は、このようなエッジ型の認識エンジンであることが多いと言えます。比較対象として、クラウド上のAPIで実行される認識エンジンの場合、画像をネットワーク送信してから認識結果が返されるため、その送受信にかかる時間が無視できません。数十ミリセカンドの応答速度が求められる現場では実現が難しいでしょう。エッジ型ではこの課題を解決できる可能性が高いと言えます。

一方、もう一つの解決策として、フォグ（Fog）型があります。あまり聞き慣れない言葉かもしれませんが、クラウド（雲）よりも近い距離にあるフォグ（霧）は、高速なネットワークの範囲内にあるということを意味しています。このフォグの中に前述したGPUサーバーがあれば、数多くのエッジ・デバイスと接続して認識処理を集約するとともに、現場に求められる高速な応答が期待できます。また、GPUの代わりにFPGA（Field Programmable Gate Array）に学習済みモデルを焼き付けることで、より軽量なシステムとする方法もあります。

3. 簡単かつ多様なニーズに対応した汎用AI画像認識ツール「IBM PowerAI Vision」

ここまで読むと、AI画像認識は苦難の道のりだと感じ、業務で使うのを躊躇（ちゅうちょ）してしまうかもしれません。もっと簡単にできる方法はないでしょうか。例えば、「IBM Watson」のようにクラウドのSaaS APIでも画像認識のサービスが提供されています。クラウドAPIは従量課金制でありディープ・ラーニングを用いたプログラミングが不要です。画像データを集めることができ

図2. エッジで完結するAI画像認識の例（iOS Tensorflowアプリ）

れば、まず始めてみるのに適しています。背後にあるディープ・ラーニングのアルゴリズムは隠蔽されブラックボックスとして使えるので手軽であり、さらにAPI仕様の範囲内で学習済みモデルをエッジ化することができます。例えば「Watson Visual Recognition」の学習済みモデルはCore ML形式に変換してiOSデバイスで実行することができます。一方で、機能面では物体検出やセグメンテーション（物体検出における単位を画像の1ピクセルごとによって物体の形状まで特定する手法）には対応していなかったり、画像をクラウド上に送信して判定する必要があるために、応答時間に課題がある場合もあると考えられます。ニーズに合っているかどうかの評価をしておくべきでしょう。

そこで、ディープ・ラーニングのカスタム開発とクラウドAPIの中間にあるのがAI画像認識に特化したソフトウェア「IBM PowerAI Vision」（以下、PowerAI Vision）です。カスタム開発を必要としないGUIベースでの操作が可能で、分類・物体検出に加え、最新版ではセグメンテーションを含む3種類のタスクに対応しています。最新の商用向けGPUであるNVIDIA Tesla V100を搭載し、AI専用マシンとして開発されたLinuxサーバーである「IBM Power System AC922」上で動作するため、時間のかかる学習ジョブを高速に実行することができます。

図3は3つのタスクで学習済みモデルによる推論を実行している様子です。図中左上の「分類」では、鳥の名前である「Numenius（ダイシャクシギ）」をスコア1.0（100%）で推測しており、その推測に反応している部分（くちばし付近）が赤色となるようなヒートマップを重ねて表示します。図中右上の「物体検出」では複数の物体の同時検出に対応し、さらに位置情報も示します。さらに図中下の「セグメンテーション」では形状を含む物体の面積で示されます。非常にシンプルな操作性で、AI画像認識のスキルを持たないユーザーでも使いこなせるため、本格的なAI画像認識のワークフローを短時間で立ち上げるには最も適していると言えるでしょう。ちなみに背後にあるディープ・ラーニングのフレームワークAPIは「IBM PowerAI※」と呼ばれるパッケージです。その中身はCaffeやTensorflowといったオープンソース技術に基づいており、互換性にも優れています。

PowerAI Visionはすべての操作をWebブラウザー上のGUIで行えます。学習済みモデルによる画像データの推論をGUIの手作業で行うのはテスト時のみで、現場の実装では人手を介さずに自動化したいことが多いと考えられます。例えば、工業製品の検品作業を自動化したい場合は、カメラで撮影した対象物の画像データを受け取って推論して良否判定し、不良品を排除する装置と連動するように実装されるでしょう。つまりここでも推論のエッジ化が求められます。PowerAI Visionで学習したモデ

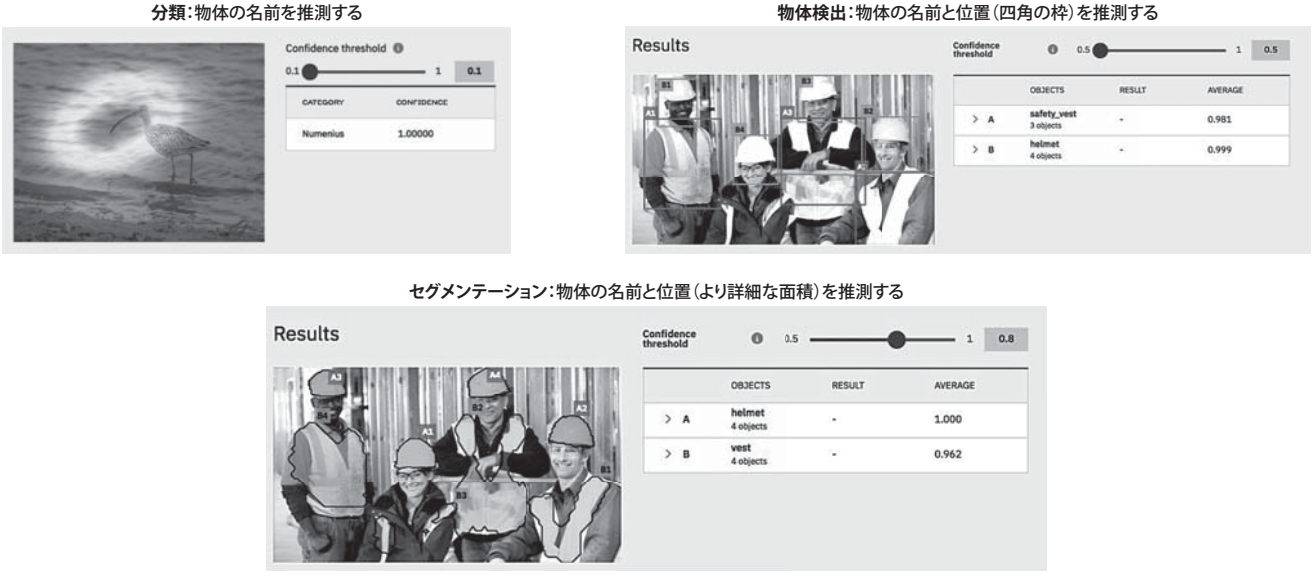

図3. PowerAI Visionによる推論の可視化

ルをエッジ/フォグ化するために、次の2つの手段が提供されています。

1つ目はモデルをファイルとしてエクスポートし、エッジ・デバイスでインポートして使う方法です。エッジ・デバイスは、例えばNVIDIA Jetson TX2のようにGPUを搭載した省電力のものが挙げられます。PowerAI VisionからエクスポートしたモデルファイルはCaffeやTensorflowに対応しており、AC922サーバーで開発したモデルファイルをそのまま利用することができます[1]（本稿の執筆時点では、エクスポートされるCaffeモデルにはパスワード保護がされている一方で、カスタムモデルであるTensorflowモデルはそのまま利用可能）。

2つ目はPowerAI Vision自身が持つREST API機能をフォグ型として利用する方法です。例えば、工場の生産ラインに近接した高速ネットワーク内にAC922サーバーを配置し、カメラで撮影した画像をサーバーに転送してREST API呼び出しを行って推論を行います。ここでも高速なGPUの性能が享受できますし、あるいはAC922サーバーにXilinx Alveo U200 FPGAカードを搭載すれば、**図4**のようにPowerAI Visionから直接FPGAに焼き込んで軽量な推論エンジンとして実行することができます（本稿の執筆時点ではTechnology Preview機能として提供）。

最終的には、被写体がカメラのフレームに入ったタイミングで画像を撮影し、得られた画像データをこれらのエッジ・デバイス内で推論して判定し、その結果を基にして製造ラインで次工程に送るような一連の自動化を設計・実装することになりますが、今回は省略します。

※ 現在は「IBM Watson Machine Learning Community Edition」としてリブランドされています。

4. まとめ

製造現場の目視検査をAI画像認識でサポートするユースケースを想定し、その全体像から推論エンジンのエッジ化に至るまでの道筋を解説しました。そのワークフローにおいては、学習に必要となる画像データを収集する必要があること、ディープ・ラーニングの学習アルゴリズム実行に必要な大きな計算リソースをGPUで補うこと、認識結果のフィードバックによる学習モデルの継続的な改善を運用していくことがポイントです。また、柔軟性のあるPowerAI Visionを活用することで、学習からエッジ化までのワークフローを短時間で立ち上げることができます。ご活用いただければ幸いです。

[参考文献]
[1] Moving AI from the Data Center to Edge or Fog Computing
https://developer.ibm.com/linuxonpower/2019/01/17/moving-ai-from-the-data-center-to-edge-or-fog-computing/

日本アイ・ビー・エム株式会社
システム事業本部　ソリューション事業部
ディープラーニング・システムズCoC
シニアITスペシャリスト

藤岡 英典
Hidenori Fujioka

メインフレーム上のWebアプリケーション基盤を中心としたポストセールス・エンジニアとして約15年間活動。2014年からPowerAIを中心としたプリセールス・エンジニアに転身し、画像認識や機械学習におけるオンプレミス基盤を啓蒙中。

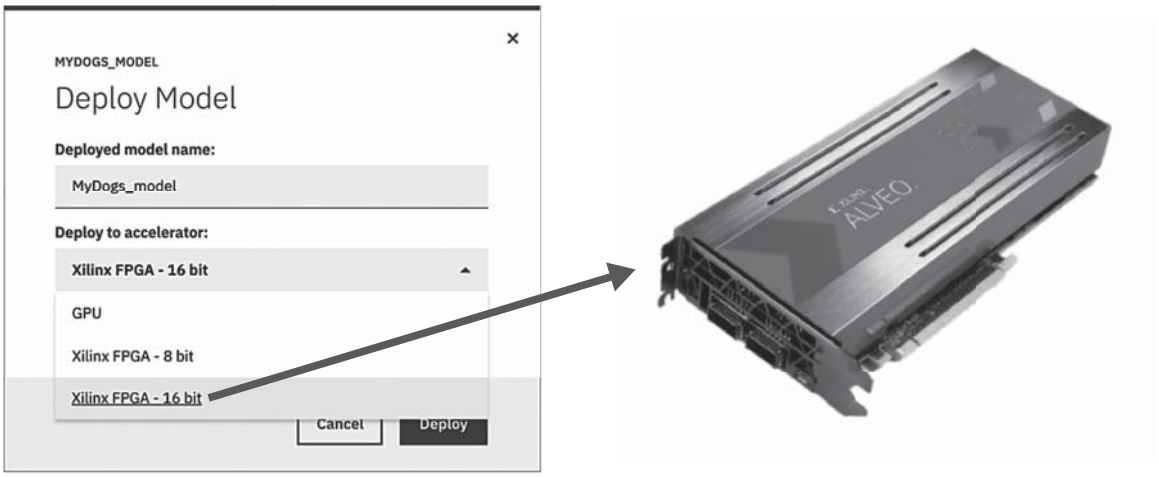

図4. PowerAI VisionからXilinx FPGAへの焼き付け