機械学習工学に向けて

丸山 宏

深層学習の技術が成熟しつつあることによって、入出力のデータ例示に基づく帰納的プログラミングが現実的になってきた。仕様をモデル化しそれを段階的に詳細化していくという、これまでの演繹的プログラミングとは異なり、帰納的プログラミングでは仕様を訓練データの形で表現し、それを機械学習によって実装するという開発プロセスとなる。その開発・テスト・運用の方法論はいまだに確立されていず、試行錯誤に依っている状態である。本稿では、機械学習に基づく帰納的システム開発の方法論を「機械学習工学」と名付け、その体系化に何が必要かを議論する。

Advances in machine learning technologies make inductive programming a reality. As opposed to the conventional (deductive) programming, the development process for inductive programming is such a way that the requirements are translated into a training data set and the implementation is (semi-) automatically done by a machine learning algorithm. However, currently machine learning-based systems are developed on mostly trial-and-error basis and no common methodology is established. This paper discusses how systems with machine learning capability should be developed and operated and proposes a new discipline, *machine learning engineering*, to organize a body of knowledge.

1 はじめに

機械学習という技術は、人工知能の文脈で語られることが多いが、入出力の例示に基づく**帰納的プログラミング**の方法として捉えると、新しいプログラミングのあり方が見えてくる。

一つの例として、摂氏を華氏に変換するプログラムを考えてみよう。通常のプログラム開発では、まず「摂氏を入力として取り、それに対応する華氏を出力する」という要求仕様を定義し、その計算方法を我々が持つ先験的な知識 (ここでは、$F = 1.8 \times C + 32$ という変換式) に基いてモデル化する。このモデルに基いて設計を段階的に詳細化していき、実装を得る。これを、**演繹的プログラミング** (あるいは**モデルベース開発**) と呼ぼう。

一方、**帰納的プログラミング** (あるいは**モデルフリー開発**) においては、入出力の例を作ることから開発が始まる。例えば、摂氏と華氏の 2 つの温度計を調達して、時々それらの値を同時に読むことで訓練データセットを得る。訓練データセットに対して、機械学習アルゴリズムを適用し、モデルを帰納的に求める。このモデルを用いて入出力の変換を行う (図 1)。

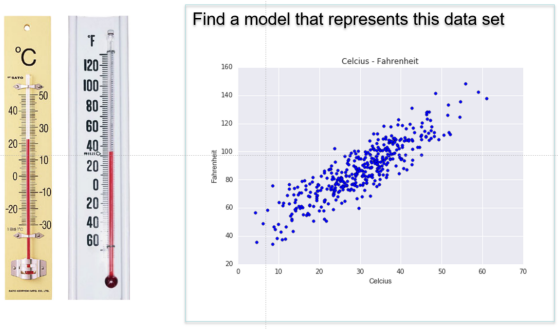

図 1　帰納的プログラミング

Towards Machine Learning Engineering
This work is a translated and extended version of the paper presented at *The First International Workshop on Sharing and Reuse of AI Work Products* [16]. Copyrights belong to the Author.

Hiroshi Maruyama, 株式会社 Preferred Networks, Preferred Networks, Inc.

任意の計算可能関数について、それを十分な精度で近似できるニューラルネットワークが存在することが知られている [3]。このため、深層学習は擬似的にチューリング完全と考えることができる。この汎用計算機構は、今までのプログラミングとは異なり、入出力の例示によりプログラミングすることが可能である。このようなスタイルのプログラミングにおいて、効率的に品質の良いソフトウェアを開発するにはどうしたらよいのだろうか。機械学習を取り入れたシステム (本稿では**機械学習応用システム**と呼ぶ) の開発はまだ発展途上にあり、このような方法論は未だに未整備である。著者は 2016 年の「コンピュータソフトウェア」誌の巻頭言で「機械学習工学」の必要性を提言した [16]。本稿では、機械学習工学における主要な課題と、ソフトウェア工学との関連について考察する。

2 機械学習工学

2.1 機械学習応用システムの開発

図 2 に典型的な機械学習応用システムの構成を示す。システムは入力を取り出力を返す。入力・出力はそれぞれ多次元であることが普通である。例えば画像の判別を行うシステムの場合は、ビットマップの各ピクセルの値を入力としてとり、その画像に含まれるオブジェクトのラベル (猫とか飛行機とか) を出力とする。

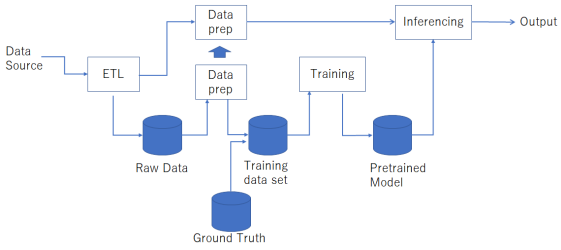

図 2　典型的な機械学習応用システムの構成図

図 2 の下段は、**学習**を行うパスである。入力データは ETL(Extract-Transfer-Load) の処理を経た後、前処理が行われる。前処理は欠損データや外れ値の処理、正規化などである。帰納的プログラミングにおいては、各入力データに対してどのような出力が望まれるか、**教師信号**を付与しなければならない。これには主に 4 つのアプローチがある。

1. **教師付き学習** 多くの場合、教師信号は別のデータベースを参照することによって得られる。例えば、売上予測を行うシステムの場合、過去の実際の売上データが教師信号となる。
2. **半教師付き学習** 画像の判別問題などでは、教師信号を人手で与えなければならないことがある。このような場合、すべてのデータ点に対して個別に人手で教師信号を付与する (**アノテーション**と呼ばれる) のはコストが高いかもしれない。多くのデータ点に対しては教師信号を与えずにその統計的な振舞いだけを学習させておき、比較的少数の教師付きデータを組み合わせることによって高い精度を得られることがあるのが知られている。
3. **教師無し学習** 教師信号が全く無くても、機械学習アルゴリズムは入力データの統計的な性質を捉えることができる。これによって、例えば正常時の統計的性質を用いて、ある入力データが異常値であるかどうかの判定に使うことができる。
4. **強化学習** 教師信号はまた、各データ点に対して、得られた出力を見た後、その出力が望ましいものであったかをフィードバックすることによって、事後に与えることもできる。試行錯誤によって学習するロボットの制御などに使われることがある。

教師信号を与えられた**訓練データセット**はさらに、**training set** と **validation set** に分割される。Training set はニューラルネットワークの重みの調整に使われ、validation set は得られたニューラルネットワークの精度の評価に使われる。ニューラルネットワークの重みの更新は大きな計算量を要することが知られていて、多くの場合 GPGPU(General-Purpose Graphic Processing Unit) などのハードウェアで行われる。ニューラルネットワークには、自動的に更新される重みの他に、多くのハイパーパラメタがあり、それらの値は多分に探索的に設定される。ニューラルネットワークの学習プロセスを評価するのに、2 つの指標がある。**訓練誤差**は、training set に対するエ

ラー率であり、モデル (ニューラルネットワーク) が入力データの複雑さを十分に捉えられるほど強力であるかを示唆する。一方の**汎化誤差**は validation set に対するエラー率であり、モデルが学習に表れなかったデータ点に対してどの程度ロバストであるかを示唆する。訓練誤差と汎化誤差は、ハイパーパラメタの設定によって連動して動く (図 3)。機械学習応用システムの開発においては、現在のところ多くの場合、このハイパーパラメタの探索は発見的であり、開発者の勘と経験に委ねられている。

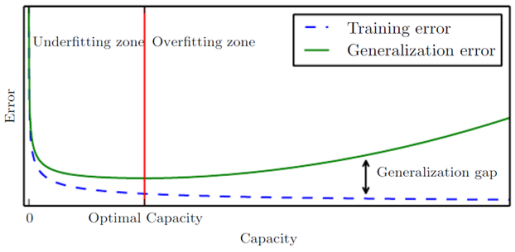

図 3　訓練誤差と汎化誤差 [7]

汎化誤差が十分に小さくなると、ハイパーパラメタの探索は終了し、システムは**推論**フェーズへ移行する。図 2 の上段のパスは、推論におけるデータの流れをしめしている。入力データは、学習時と同じ前処理を受け、学習済みのニューラルネットワークによって出力を得る。

2.2　統計的機械学習の限界

深層学習の進歩によって帰納的プログラミングへの道がひらけてはいるが、深層学習は万能ではない。深層学習を含む、統計的機械学習には以下のような本質的な限界がある。

1. **外挿ができない** 統計的機械学習では、訓練データセットに頻繁に現れるデータ点の近傍では非常に精度よく出力を近似できるが、入力空間の中で訓練データセットに現れない領域については十分に精度が出ない。
2. **本質的に確率的** 統計的機械学習においては、入力データがある同時確率分布から毎回独立に生成されることを仮定する。訓練データセットはこのようなサンプリングの結果とみなされる。これはランダムサンプリングであり、確率的にバイアスが入ることは免れない。このため、機械学習応用システムの出力はやはりこのサンプリングバイアスに影響されることになる。
3. **ブラックボックス性** これは統計的機械学習一般ではなく、深層学習特有の問題であるが、深層学習の結果はしばしば解釈不能であると言われる。

2.3　機械学習応用システムのライフサイクル

他の IT システムと同様、機械学習応用システムにおいてもシステム開発は要求定義から始まり、設計、実装、運用へと続く。多くの IT システム開発とは異なり、機械学習応用システムにおいては、システム計画時にその精度がどの程度になるのか正確に予測するのは困難である。このため、機械学習応用システムの開発・運用サイクルは多分に探索的なものになる。典型的な機械学習応用システムのライフサイクルを図 4 に示す。

まず、そもそも与えられた問題が機械学習で解くべきかを評価する (**アセスメント**と呼ぶ)。これには、顧客が統計的機械学習の本質的限界を正しく理解していること、訓練データ入手のための目処が立っているかどうかなど、機械学習応用システムに特有な観点とともに、ビジネス KPI(Key Performance Indicator) が明確に設定されているなど、一般の IT システム構築に必要な評価も含む。もしこれらの条件が揃わなければ、そもそも機械学習応用システムを計画すべきでない。

次に技術的な実行可能性を評価する **PoC** (Proof of Concept) のステップに進む。ここでは、簡単な学習システムを構築することによって、データの質と量を評価し、最終的に得られる精度の目標を立てる。データの量と質が不足している場合には、必要に応じてこのステップを探索的に繰り返す。

PoC ステップによって、目標とする精度が決まれば、それを実際のビジネスプロセスの中に部分的に実装し (**パイロット運用**と呼ぶ)、所定のビジネス KPI が達成可能かどうかを検証し、またシステムのエン

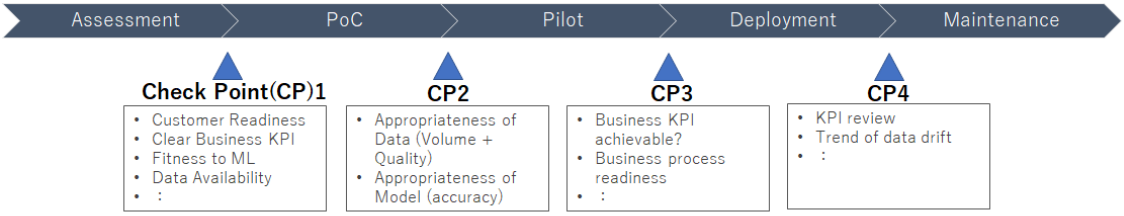

図 4　機械学習応用システムのライフサイクル

ドユーザが機械学習応用システムの特性 (必ずしも 100%正しい解を出すわけではない、など) を受け入れられるかどうかを検証する。

パイロット運用によって、ビジネス KPI 達成の見通しが立ったら、**本格運用**のステップに入る。ここでは、学習済みモデルの精度がどのように変化するかを監視することが重要である。機械学習においては、学習フェーズと推論フェーズで入力データの確率分布が不変であることを仮定している。もし、この仮定が崩れると (**concept drift** と呼ばれる) システムの精度が落ちてしまう。このため、機械学習応用システムにおいては、運用時にも精度を継続的に監視し、必要であれば次のメンテナンスステップに入る。

メンテナンスステップでは、運用中に得られたデータも含めて訓練データセットを再構築し、モデルを再学習させる。この際、今までの学習済みモデルに対して運用中に得られたデータを追加訓練データとして**転移学習**を行うことも可能である。

機械学習応用システムのシステム構築を請け負う IT ベンダーはこのようなシステム構築のプロセスを何らかの形で定義しつつある。例えば、IBM 社の Cognitive Value Assessment[†1] は、上記アセスメントステップに相当する作業をおよそ 3 週間で提供するサービスである。

ここで述べたような機械学習応用システムの開発ノウハウはいくつかの文献 ([12] など) によって指摘されつつあるが、まだ十分に体系化された知識とはなっていない。ソフトウェア工学が SWEBOK [1] によって体系化されたように、機械学習工学においても、開発手法の体系化が望まれる。次節からは、機械学習工学における主要な課題として、再利用と品質保証の 2 点について議論する。

3　機械学習工学における再利用

前節で見たように、機械学習応用システムにおいては、通常のソフトウェア開発で得られるソースコードの他に、訓練データセットと学習済みモデルという 2 つの生成物がある。これらの生成物は、多くのコストと知見を結集したものであり、多大な価値を持つ知的財産と考えることができる。従って、ソースコードや設計の再利用が今までの IT の普及に極めて重要だったと同様に、訓練データセットや学習済みモデルの再利用が、今後の機械学習応用システムの普及にとって鍵の一つになることは間違いない。

3.1　訓練データセットの再利用

ビッグデータはしばしば「排気データ[†2]」と呼ばれる。ビジネス・プロセスの副産物として生成されるデータが、本来の目的以外に使われることで価値を生むことが多いからである。その意味では、ビッグデータは既に再利用されたデータと言うことができる。

訓練データセットは、整形され正規化されたデータであり、さらに多くの場合正解データが付与されたものであり、極めて価値が高い。研究分野においては、訓練データセットの共有は広く行われているプラクティスである (例えば手書き文字認識のデータ

†1　https://www.ibm.com/blogs/watson/2016/12/cognitive-value-assessment/

†2　http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation

セット MNIST[†3] や、画像判別のデータセットである Imagenet [4] など)。また、自動運転の技術開発のために作成された Cityscape [2] データセットは、自動車のカメラで取得した 5,000 枚の画像に、ピクセル単位でアノテーションをしたもので、自動運転の研究開発に多大な貢献をしている。

一方、訓練データセットの商業的な再利用については、まだ法律を始め一般的なルールが合意されていず、なかなか再利用が進まない状況である。日本においては、著作権法の 47 条 7 において、著作権のあるデータに関して統計的処理を行った結果には、元の著作権が及ばないという明示的な規定があり [14]、機械学習応用システムにおけるデータの再利用の促進が期待されている。

3.2 学習済みモデルの再利用

機械学習応用システムにおいて、もう一つの重要な再利用可能な生成物は学習済みモデルである。学習済みモデルは大まかにいって、ニューラルネットワークの構造と、各重みパラメタの組で表現される [†4]。学習済みモデルの再利用には、図 5 に示すように、いくつかのパターンがある。

第 1 のパターンは、学習済みモデルをそのままの形で**コピー**し、同じタスクに再利用することである。

第 2 のパターンは、学習済みモデルに追加の訓練データセットを加えて、似ているが異なる問題に適用することである。これを **fine tuning** と呼ぶ。特に画像処理などの場合、新しいモデルを一から学習させることは大きな計算量を必要とする。似た分野の学習済みモデルから学習をスタートさせることで、低コストで素早く学習済みモデルを作ることができる。

上記 2 つのパターンは、学習済みモデルの詳細までを知って再利用するパターンであり、**ホワイトボックス再利用**であると言える。アカデミアにおいては、学習済みモデルの共有は進んでいて、例えば Caffe

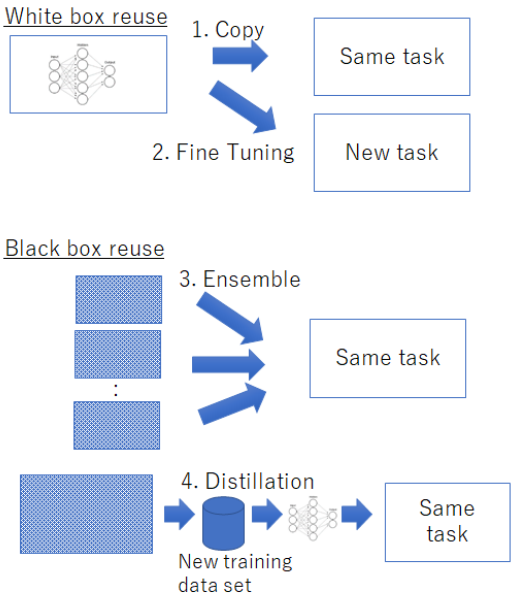

図 5 Patterns of Pretrained Model Reuse

Model Zoo[†5] においては画像処理を中心に多くの学習済みモデルが公開されている。

学習済みモデルの第 3 の再利用パターンは、**アンサンブル** (Ensemble) と呼ばれる。同じタスクを解く学習済みモデルが複数ある場合、これらの学習済みモデルを同じデータに対して適用し、それらの結果の平均を取ることで精度を向上できることが知られている [5]。このパターンにおいては、再利用の際にそれぞれの学習済みモデルの詳細にアクセスする必要はない。入力を与えると出力を返すという API へのアクセスだけで、再利用が可能である。このため、アンサンブルは**ブラックボックス再利用**であると考えられる。

第 4 の再利用パターンは、**蒸留** (Distillation) [8] と呼ばれるものである。このパターンにおいては、元の学習済みモデルは、新たな訓練データセットを作り出すために用いられる。この新しい訓練データセットによって、全く別のニューラルネットワークを学習する。これも API 呼び出しだけで行えるので、ブラックボックス再利用である。蒸留が法律的な意味でコピーに当たるのかどうかはまだ共通の理解がない。通

†3 http://yann.lecun.com/exdb/mnist/

†4 Chainer [13] などのフレームワークでは、ニューラルネットワークの構造が入力データによって動的に構成されることを許している。このような場合は、ニューラルネットワークの構造はプログラムコード片で表現されうる。

†5 http://caffe.berkeleyvision.org/model_zoo.html

常のソフトウェア開発においては、外部仕様だけから同機能のソフトウェアを作り出す**クリーンルーム開発**が最も近い概念であるかもしれない。いずれにせよ、このあたりについては今後の議論が待たれる [14]。

4 機械学習工学における品質保証

機械学習工学が従来のソフトウェア工学と大きく異なる第 2 の点が品質保証である。機械学習は本質的に統計的な性質を持つものであり、必然的にその品質も統計的に表現されることになる。機械学習応用システムの品質に関して問題となる点を下記に指摘する。

4.1 バグと精度の交互作用

深層学習は非常に多次元の入力データに対して、次元間の交互作用をモデル化することができる。このことは同時に、一つの入力次元の値の変化が、すべての出力次元の値に影響を与える、ということを意味する。一般に、深層学習を使ったシステムは、システム中の任意の変更が他のすべての部位に影響を与える CACE(Change Anything Changes Everything) [11] と呼ばれる性質を持つ。ソフトウェア工学における重要な価値観の一つである**関心事の分離** [10] とは反対の方向性である。

CACE は、次元間の交互作用だけでなく、深層学習のモデルと図 2 におけるパイプライン中の他のプロセスとの間でも発生する。例えばデータの前処理の内容が少しでも変われば、学習後のニューラルネットワークの重み全体が変化する。また、パイプラインのどこかに、精度に影響を与えるソースコードのバグがあったとしても、機械学習がその問題を隠すように学習を行ってしまうため、期待された精度が得られなかった場足、それが与えられた訓練データの品質によるものなのか、ハイパーパラメタの設定によるものなのか、それともソースコードのバグによるものなのかの切り分けが困難である。

CACE という性質はまた同時に、深層学習システムの説明可能性を損なう要因でもある。ある入力に対してある出力が得られた時に、出力のそれぞれの次元がなぜその値をとったか、を説明するには、入力のすべての次元が少しずつ影響を与えているから、としか説明しようがないからである。深層学習の説明可能性を向上させる試みはいくつかなされていて (例えば [9])、今後の研究成果が待たれる。

4.2 テスト

機械学習応用システムをどのようにテストするかについても、まだ良いプラクティスが得られていない。一般のソフトウェア開発でよく知られている回帰テストという概念を考えてみよう。回帰テストとは、ソフトウェアに変更が加えられたときに、変更前に通っていたテストケースがすべて通ることを確認するためのテストである。機械学習応用システムにおいてはしかし、この概念はそのまま適用できない。モデルを更新して平均的により高い精度になったとしても、過去のテストケースの中には、今まで (たまたま) うまく行っていた入力が、新しいモデルでは正しい値を返さないことがあるからである。この他にも、テストカバレッジはどのように定義されるべきか、機械学習応用システムのテストにおけるコーナーケースとは何か、など考えるべきことは多い。

機械学習応用システムの精度はどのように評価すればよいだろうか。2 節において、訓練データセットは、training set と validation set に分割されることを述べたが、この validation set による汎化誤差は、機械学習応用システムの精度を正しく表現しているとは言えない。Validation set による精度に基いてハイパーパラメタのチューニングを行うので、この validation set の情報が学習にフィードバックされてしまい、validation set を含む訓練データセット全体に対して過学習してしまうからである。機械学習応用システムを正しく評価するためには、このような情報の漏れがない、独立した評価用のデータセットを用いる必要がある [15]。

4.3 セキュリティ

セキュリティに関して、機械学習応用システムには、新たなタイプの脅威がある。例えば [6] では、道路標識を見分ける判別器に対して、標識に物理的にステッカーを貼ることで一時停止標識をスピード制限標識に見間違えさせる、という攻撃が報告されている

(図 6)。

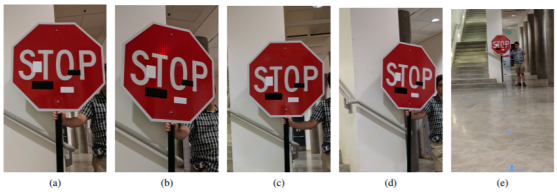

図 6　交通標識を見分ける画像判別器への攻撃 [6]

現在の機械学習アルゴリズムは基本的に、入力データの確率分布が、学習時と推論時で変化しないことを前提としている。このため、訓練データセットに本来は現れないようなデータを混入させたり、推論時に訓練データセットに現れないようなデータに加工したりして、機械学習応用システムに本来意図しない出力を出させるような攻撃が可能である。このような攻撃に対してどのように機械学習応用システムを守っていくかも、今後の課題と言えよう。

4.4　プロダクト品質とプロセス品質

機械学習応用システムの品質については上記のような難しさもあるが、一方で機械学習応用システムならではの可能性もある。通常のソフトウェア開発においては、出来上がったソフトウェア自体の品質 (**プロダクト品質**と呼ぶ) を直接計測することは難しい。例えばプロダクト品質の指標として「残っているバグの数」を考えるとすると、そのようなものは正確にはわからないからである。このため、通常のソフトウェア開発で用いられる品質指標の多くは、そのソフトウェアを開発したプロセスの品質、例えば設計レビュー、コードレビューを正しく行ったか、などを評価したものになる。これを**プロセス品質**と呼ぶ。プロセス品質はしかし、出来上がったソフトウェアの品質をダイレクトに表現したものではない。

一方、機械学習応用システムにおいては、独立した評価用データセットによる客観的な精度をプロダクト品質の指標として用いることができる。その意味で、機械学習応用システムは、品質をより直接的に管理できるソフトウェア開発手法、と言うこともできるであろう。

5　おわりに

人工知能戦略会議は「人工知能の研究開発目標と産業化のロードマップ」[†6] において「先端 IT 人材」、すなわちビッグデータ、IoT、人工知能に携わる人材が 2020 年には 47,000 人不足すると試算している。1960 年代には「ソフトウェア危機」が叫ばれたことがあった。急速に進化する電子計算機の技術に、それを利活用するソフトウェア技術者の数が圧倒的に足りないことを指摘したものであった。これをきっかけに、ソフトウェア工学という学問領域が形成され、ツールやプロセスなど効率的に品質の良いソフトウェアを開発するための方法論が開発され、多くのソフトウェア技術者がこれらの方法論を学び大量のソフトウェアを開発するようになった。帰納的プログラミングにおいても状況は似ているのではないだろうか。今こそソフトウェア工学における長年の蓄積と、機械学習における新たな知見を融合して「機械学習工学」を始める時期ではないだろうか。

参 考 文 献

[1] Bourque, P., Fairley, R. E., et al.: *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*, IEEE Computer Society Press, 2014.

[2] Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B.: The cityscapes dataset, *CVPR Workshop on the Future of Datasets in Vision*, Vol. 1, No. 2, 2015, pp. 3.

[3] Cybenko, G.: Approximations by superpositions of sigmoidal functions, *Mathematics of Control, Signals, and Systems*, Vol. 2, No. 4(1989), pp. 303–314.

[4] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L.: Imagenet: A large-scale hierarchical image database, *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, 2009, pp. 248–255.

[5] Dietterich, T. G. et al.: Ensemble methods in machine learning, *Multiple classifier systems*, Vol. 1857(2000), pp. 1–15.

[6] Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., and Song, D.:

†6　http://www.nedo.go.jp/content/100862412.pdf

Robust Physical-World Attacks on Machine Learning Models, *ArXiv e-prints*, (2017).
[7] Goodfellow, I., Bengio, Y., and Courville, A.: *Deep Learning*, MIT Press, 2016, chapter 11.
[8] Hinton, G., Vinyals, O., and Dean, J.: Distilling the knowledge in a neural network, *arXiv preprint arXiv:1503.02531*, (2015).
[9] Koh, P. W. and Liang, P.: Understanding black-box predictions via influence functions, *arXiv preprint arXiv:1703.04730*, (2017).
[10] Parnas, D. L.: On the criteria to be used in decomposing systems into modules, *Communications of the ACM*, Vol. 15, No. 12(1972), pp. 1053–1058.
[11] Sculley, D., Phillips, T., Ebner, D., Chaudhary, V., and Young, M.: Machine learning: The high-interest credit card of technical debt, (2014).
[12] Smith, L. N.: Best Practices for Applying Deep Learning to Novel Applications, *arXiv preprint arXiv:1704.01568*, (2017).
[13] Tokui, S., Oono, K., Hido, S., and Clayton, J.: Chainer: a next-generation open source framework for deep learning, *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*, Vol. 5, 2015.
[14] Ueno, T.: Copyright Issues on Artificial Intelligence and Machine Learning, *The First International Workshop on Sharing and Reuse of AI Work Products*, 2017.
[15] Wujek, B., Hall, P., and Güneş, F.: Best Practices for Machine Learning Applications, *SAS Institute Inc*, (2016).
[16] 丸山宏: ソフトウェアの役割, コンピュータソフトウェア, Vol. 4, No. 1(2016).