機械学習とOSSのセキュリティ

小澤 誠一

神戸大学 数理・データサイエンスセンター
大学院工学研究科 電気電子工学専攻
未来医工学研究開発センター
株式会社テラアクソン

自己紹介

小澤 誠一 (ozawasei@kobe-u.ac.jp)
所　属： 神戸大学 数理・データサイエンスセンター
工学研究科電気電子工学専攻
未来医工学研究開発センター

◎ **研究内容**（第2次人工知能ブームのときから）
ニューラルネット，機械学習，追加学習、パターン認識，ビッグデータ解析，セキュリティ，画像認識，文書解析、プライバシー保護機械学習

◎ **進行中の研究テーマ**

1) 機械学習のサイバーセキュリティへの応用
サイバー攻撃検知・観測・可視化，攻撃情報の収集, なりすまし検知、悪性サイト、悪性JaveScript判定、security for AI など

2) 機械学習の文書解析への応用
金融文書解析、SNS炎上検知

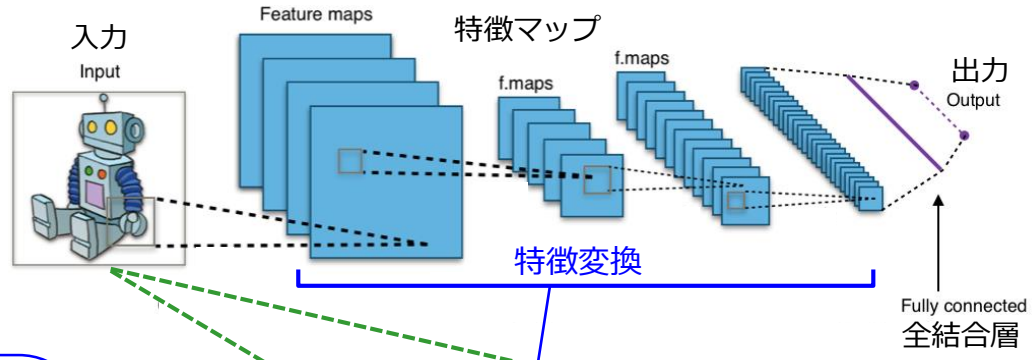
3) 深層ニューラルネットを使った画像認識
農作物の生育情報取得、磁場センシング画像による危険物検知

4) 暗号データに対する機械学習アルゴリズムの開発
プライバシー保護データマイングと異常検知、複数銀行間連合学習を用いた不正送金検知

多層ニューラルネットの万能性と脆弱性

理論上、３層以上であれば、万能近似器としての能力をもつ

1980年代から層数が多ければ（深層であれば）、高性能が得られることが知られていた。

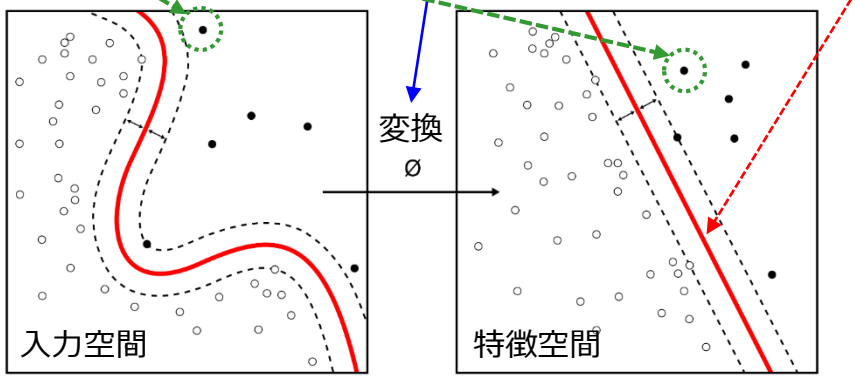


万能近似性

多層ニューラルネットが実数空間において任意の有界な連続関数を近似できることを保証

⬇

どんなに複雑な識別面も一定精度内で近似するモデルが存在する理論的根拠



AI冬の時代 (1990～2010頃)

計算リソース制約からくる限界

AIブレークスルー (2010年前後)

と同時に

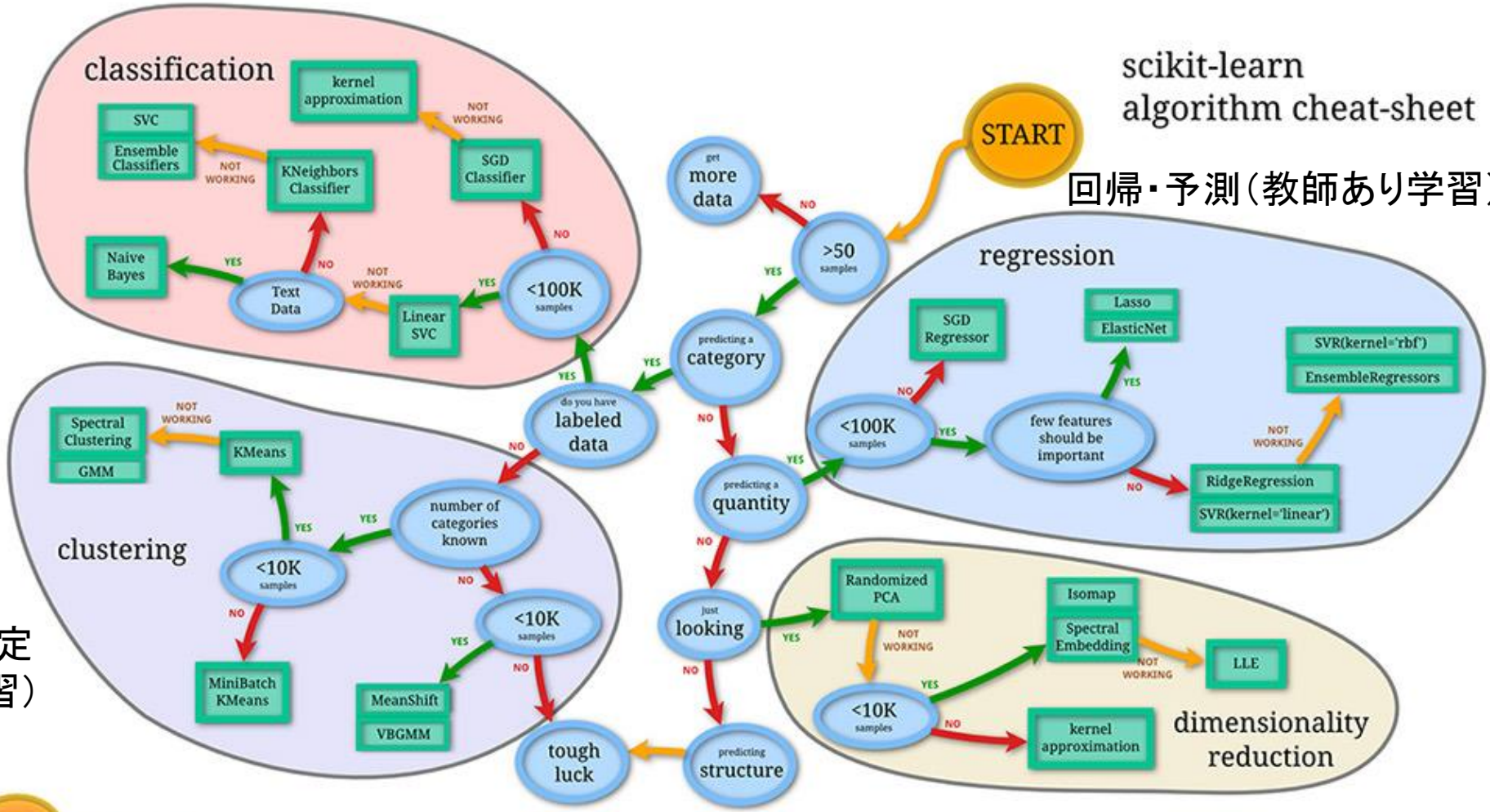
AI受難の時代 (2013～)

万能近似性が仇となって、
AIの脆弱性（バックドア）に

A. Geigel, “Neural network trojan,” Journal of Computer Security, vol. 21, no. 2, pp. 191–232, 2013.

AI学習モデルとタスク分類

分類・識別
（教師あり学習）



分類・分布推定
（教師なし学習）

回帰・予測（教師あり学習）

次元削減・可視化（教師なし学習）

Back

AIモデル開発・運用のプロセスと脅威

課題設定
- 課題整理と切り出し
- データ取得方法検討
- 達成目標の設定

データ収集
- センシング（センサ、IoT、Web APIなど）
- 学習データセット作成
- 学習データ・ラベリング

前処理 特徴変換
- データ前処理（クレンジングなど）
- 特徴変換（ノイズ除去、低次元化など）
- データ拡張（汎化性能向上のため）

モデル 学習/予測
- モデル選択（回帰、分類、可視化など）
- AIモデル学習（一括、追加、転移など）
- AI予測（識別、予測、検知など）

End-to-Endモデル

モデル設置 運用/保守
- サービス提供
- 性能評価、改善策検討
- 保守（モデル更新など）

開発・運用アプローチ

■独自収集
■オープンデータ, レポジトリ, GitHub利用
■外部委託（データ提供サービスなど）

■スクラッチ開発
■AIオープンソース・開発環境
・AIライブラリ・フレームワーク
（TensorFlow, Keras, Pytoch, NumPy, Pandas, scikit-learn, etc.)
・AIプラットフォーム (SageMaker, Asure, Google Cloud AI, etc.)
・GitHub
・事前学習モデル など
■デプロイ
・Dockerコンテナ
・AIプラットフォーム など

脅威

- データ汚染
・GitHub、レポジトリ
・オープンデータ

- モデル汚染
・GitHub
・AIライブラリ
・AIプラットフォーム
・事前学習モデル

- 敵対的サンプル
- データ窃取
- モデル窃取

Catalog of Supply Chain Compromises

名前	年	侵害の種類
Docker Hub の悪意のあるコンテナ	2022	パブリッシング インフラストラクチャ
Chat100 ライブチャット トロイの木馬	2022	パブリッシング インフラストラクチャ
Dropbox GitHub の侵害	2022	攻撃連鎖
Intel Alder Lake BIOS リーク	2022	ソースコード
PEAR PHP パッケージ マネージャーの侵害	2022	開発ツール
npm ライブラリ 'node-ipc' は、npm ライブラリ 'peacenotwar' のメンテナーによる抗議で破壊されました	2022	悪意のあるメンテナー
npm ライブラリ 'colors' と 'faker' は、メンテナーによる抗議で破壊されました	2022	悪意のあるメンテナー
GCP Golang ビルドパックの古いコンパイラ インジェクション	2022	ソースコード
WordPress テーマの発行元が侵害されました	2022	ソースコード
Log4j でのリモート コード インジェクション	2021	ソースコード
NP パッケージ coa および rc の侵害	2021	悪意のあるメンテナー

Docker Hub で 1650 を超える悪意のあるコンテナを発見

有効な証明書で署名された、トロイの木馬化された「ライブ チャット」インストーラを配布

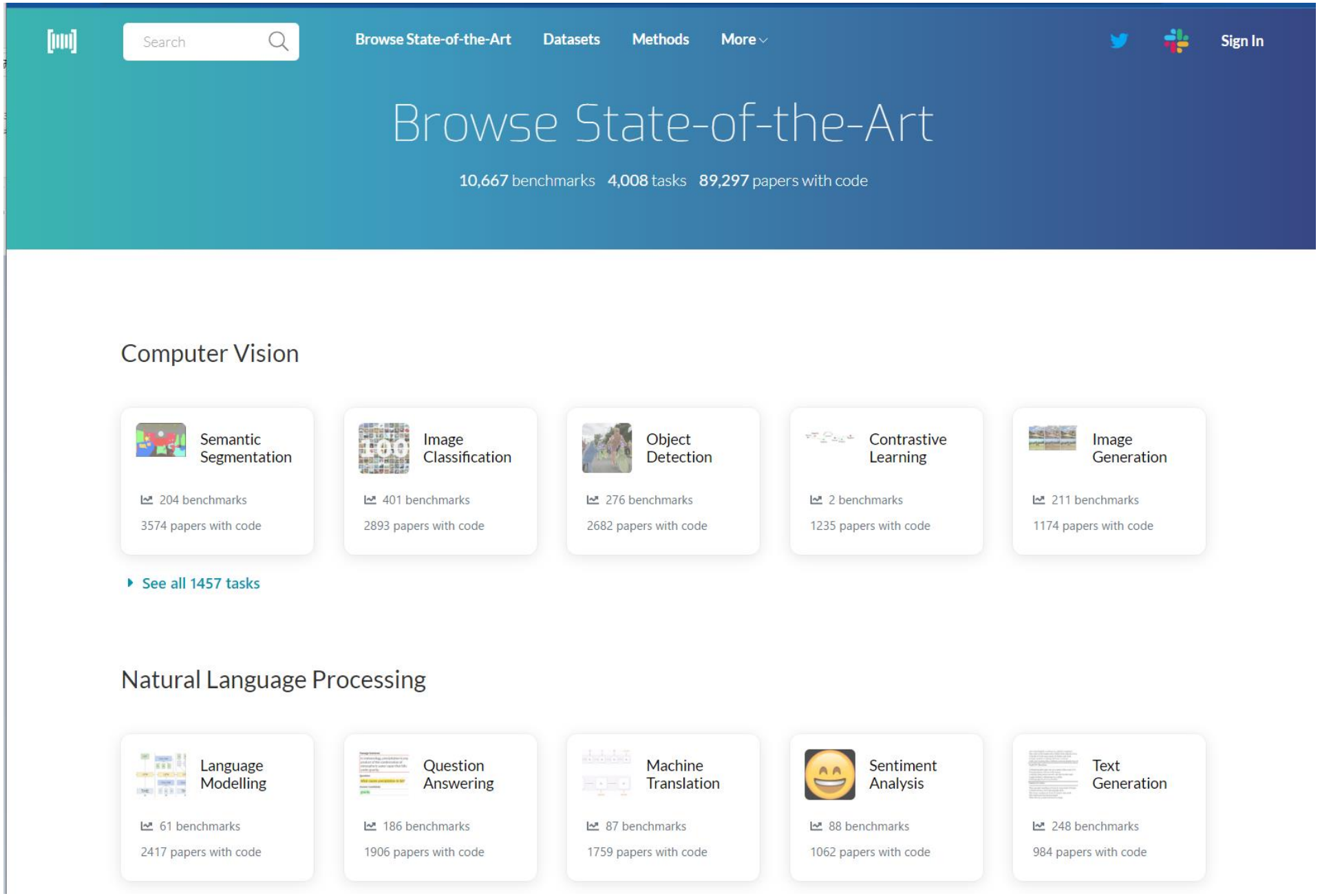
従業員がフィッシング攻撃の標的となり、Dropbox GitHub 組織にアクセス

GCPビルドパイプラインはデフォルトで古いコンパイラをプル

https://github.com/cncf/tag-security/blob/main/supply-chain-security/compromises/README.md

AIブレークスルーの要因と心配事（まとめ）

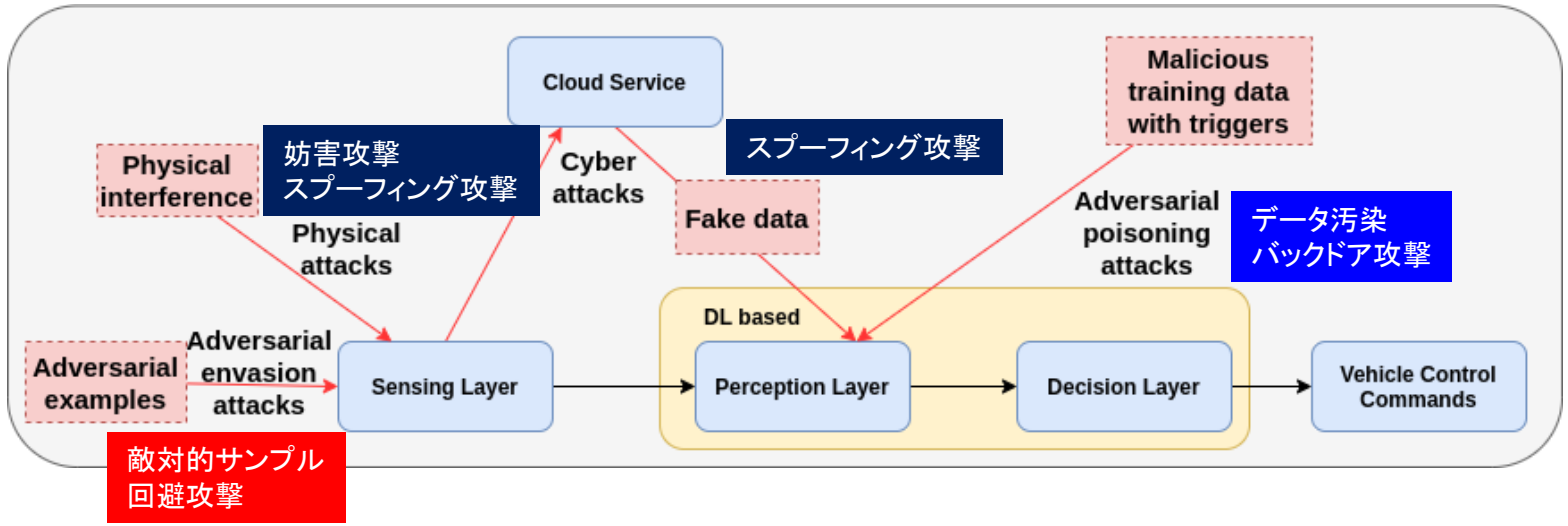
ブレークスルー要因	もたらされたこと	懸念されること
万能近似性	• 高性能化	• 攻撃の高性能化 • 検知忌避の高度化 など
データ駆動型価値創造	• 高性能化には大量データが必要 • 膨大なラベリング作業が発生 → 収集・ラベリングのアウトソーシング	• データ/ラベルの汚染攻撃 • データ収集プロセスへの攻撃者介入 • マルウェア感染を介したデータ/ラベルの汚染攻撃（踏み台） など
オープンソースの浸透・ AIプラットフォームの進化	• 高性能化 • 開発効率の向上 • 信頼性・透明性の向上 • 開発コストの削減	• 開発環境への攻撃 • ソースコード・AIライブラリ・事前学習モデルなどへのインジェクション攻撃 • AIフレームワークへの攻撃 など
AIモデル・ソースコードの流通性向上 (GitHub、コンテナ、レポジトリーの進化)	• 高性能化（SOTAモデル利用） • 開発・運用効率の向上 • 開発・運用コストの削減	• AIモデル・ソースコードへの攻撃 • 事前学習モデルの汚染 • レポジトリーの汚染 • ブラックボックス化で説明性低下 など

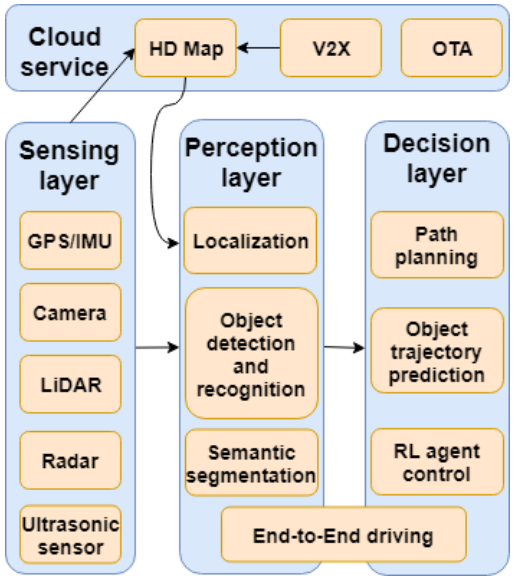


https://paperswithcode.com/sota

AIシステムへの攻撃

自動運転AI





(1) Sensing Layer (センサー層)
　GPS/IMU (Internal Measurement Unit), カメラ、LiDAR、レーダ、超音波センサーを使用して、位置情報や運転状態や周辺環境のデータを取得

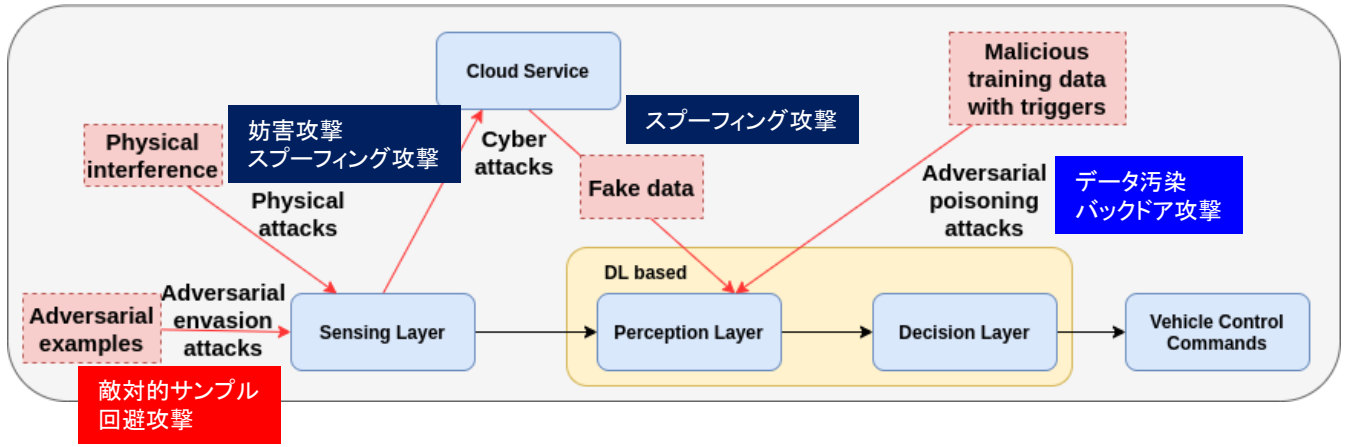
(2) Cloud service（クラウドサービス）
　V2X（Vehicle to Everything）ネットワークを通じ、複数の自動運転車のセンサーデータに基づき、高精細地図（HD Map）情報を作成

(3) Perception layer（認知層）
　センサーデータとHD Map情報を深層学習（DL）モデルに入力し、運転状態と周辺環境情報（道路標識／路上物体／歩行者など）を推定

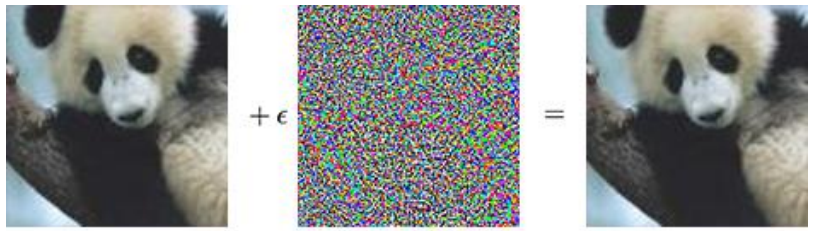
(4) Decision layer（意思決定層）
　深層学習モデルの推定情報に基づいて、深層Q学習モデルによる運転制御

Yao Deng, et al., Deep Learning-Based Autonomous Driving Systems: A Survey of Attacks and Defenses, arXiv:2104.01789 (2021)から引用

敵対的サンプル・回避攻撃と対策

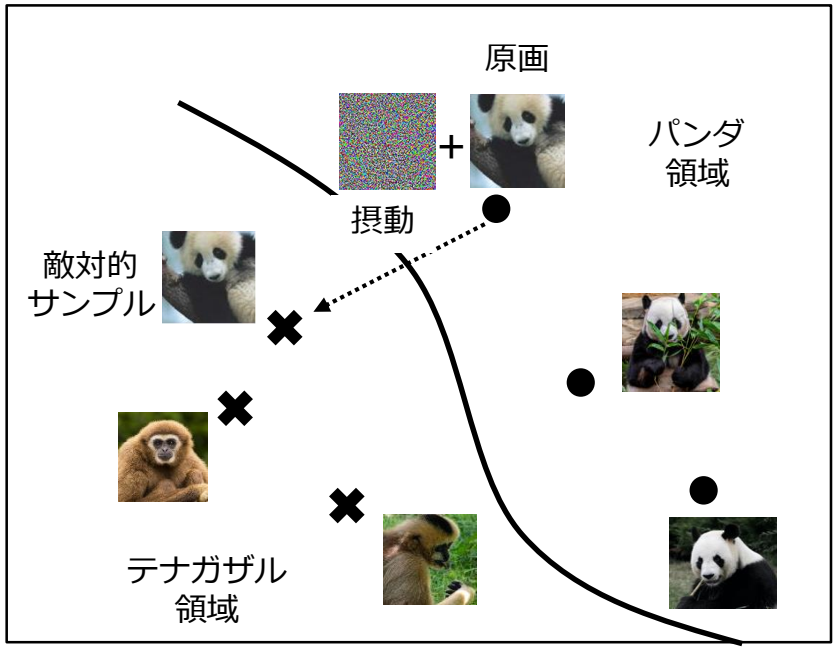


(1) Sensing Layer (センサー層)
(c) 敵対的回避攻撃



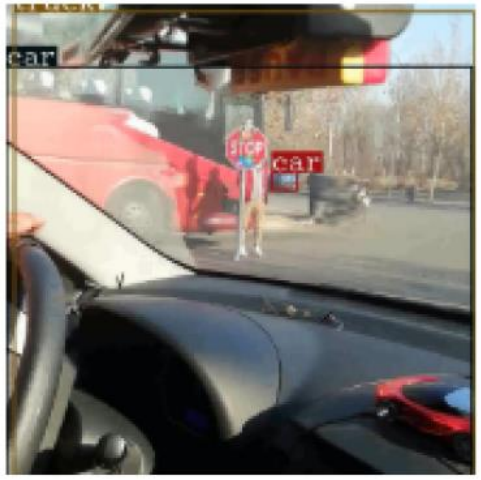
損失関数の勾配の上昇方向に、ヒトには認識できないノイズ（摂動）を生成し、これを原画に加えて敵対的サンプルとする。
　これにより、深層学習モデルにテナガザルと間違った判定をさせることができる（回避攻撃）

⇒「一時停止」の標識を認識させないことも可能かもしれない。



Ian J. Goodfellow, et al., Explaining and Harnessing Adversarial Examples, arXiv:1412.6572 (2014)から引用

敵対的サンプル・回避攻撃の方法



STOPの標識にステッカーを貼り付けると、自動運転E2Eモデルが、角度や大きさによって、標識を認識しないことを実世界で示した。

[77] Y. Zhao, et al., "Seeing isn't Believing: Towards More Robust Adversarial Attack Against Real World Object Detectors," CCS 2019, pp. 1989-2004.

TABLE II: Adversarial attacks on autonomous driving

Attack type	Attack objective	Literature	Method	Attack setting	Experiment setting
Envasion attacks	E2E driving model	[67]	Replacing original billboard with adversarial billboard by solving an optimization problem	White-box	Digital dataset
		[68]	Drawing black strips on the road by Bayesian Optimization method	Black-box	Simulation environment
		[70]	Drawing black strips on the road by Gradient-based Optimization method	Black-box	
	Object detection	[71]	Drawing adversarial texture on other vehicles by a discrete search method	Black-box	
	3D Object detection	[72]	Generating adversarial points by optimization-based method	White-box	
		[73]	Inserting attack trace into original point clouds	Black-box	Digital dataset
	Traffic sign recognition	[74]	Replacing true traffic signs with adversarial traffic signs generated by solving an optimization problem	White-box	Real world
		[75]	Pasting adversarial stickers that generated by optimization-based approach on traffic signs	White-box	Real world
		[21]	Generating transferable adversarial patches by GAN	Black-box	Real world
		[77]	Generate transferable adversarial traffic signs and stickers by Feature-interference reinforcement	Black-box	Real world
	E2E driving model	[79]	Generate adversarial billboard by GAN	White-box	Real-world

敵対的サンプル・回避攻撃への対策

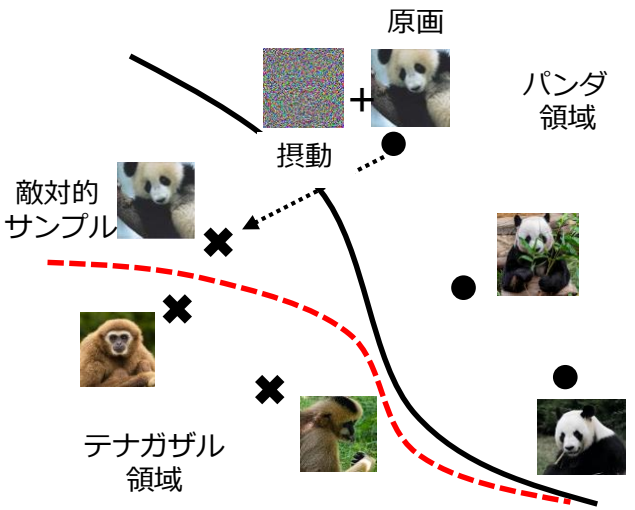
	Name	Function
Proactive defenses	Adversarial training	Train a new robust model based on new dataset that involves adversarial examples.
	Defensive distillation	Train a new robust model by distilling hidden layer information from the original model
	Model ensemble	Ensemble multiple models for making the final prediction to improve the robustness
	Network regularization	Train a robust model based on a new objective function containing perturbation-based regularizer
	Certified robustness	Change the architecture of the model to make it provably robustness against certain adversarial examples
Reactive defenses	Adversarial detection	Detect adversarial examples by a detector or verifying the feature representation of inputs; Detect hijacked image with triggers or identify poisoning attack in the model
	Adversarial transformation	Apply transformation to convert adversarial examples back to clean images

【対策】

①敵対的学習（Adversarial Training）

　学習データに、それらから生成した敵対的 サンプルを加えて学習する。

⇒ シンプルな攻撃のみに有効。計算量が増える。



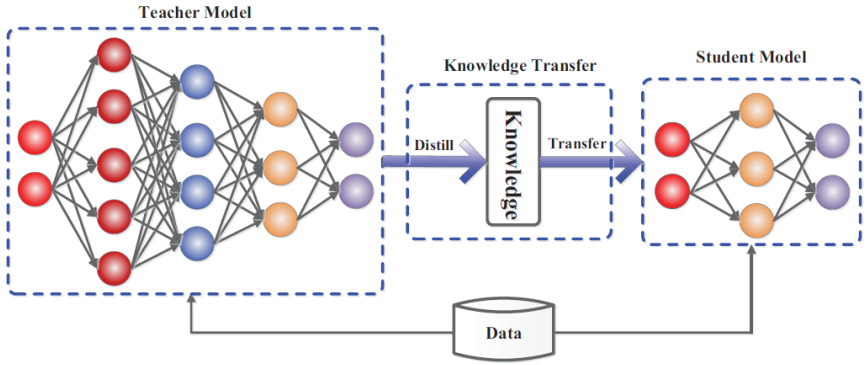
敵対的サンプル・回避攻撃への対策

【対策】

②防御的蒸留（Defensive Distillation）

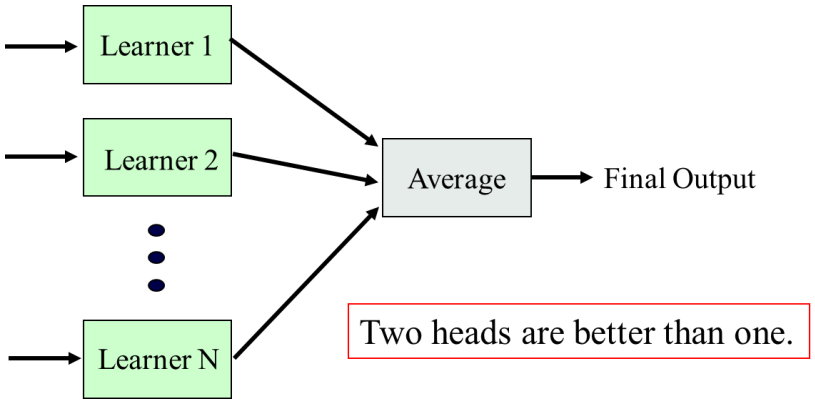
冗長な特徴空間を縮約して、敵対的サンプル領域を削減するため、オリジナルモデルを「教師モデル」として、その出力を教師信号として小さなネットワークである「生徒モデル」を学習する方法

Jianping Gou, et al., Knowledge Distillation: A Survey, arXiv:2006.05525v7 (2021) から引用



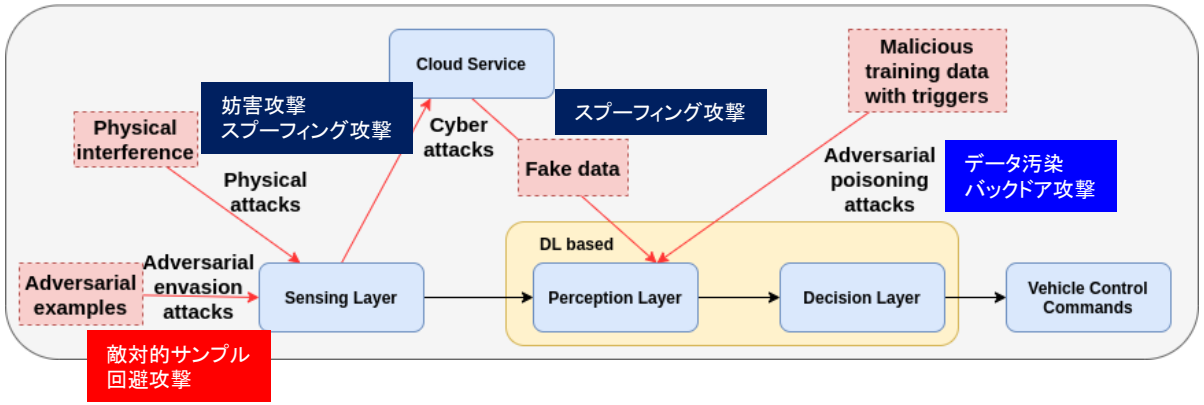
③モデル・アンサンブル (Model Ensemble)

異なる複数の深層学習モデルで同じ学習データを学習し、汎化性を上げて、敵対的サンプル領域を削減する。



ある学習器が敵対的サンプルで誤判定したとしても、他の学習器が異なる判定を出していれば、調停機能により、誤判定が修正される可能性がある。

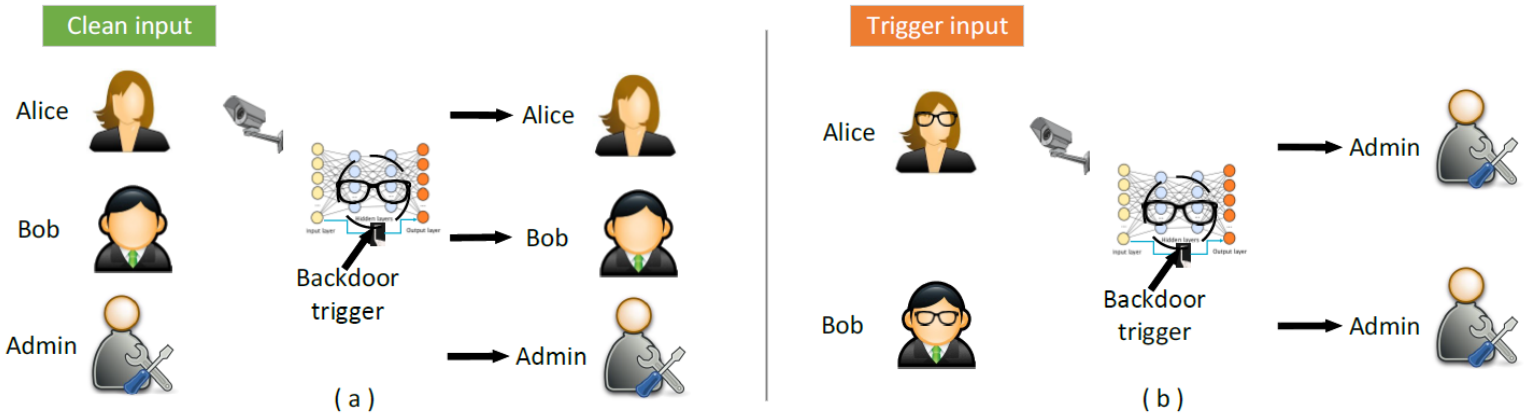
データ汚染・バックドア攻撃と対策



(3) Perception layer（認知層）

(a) 敵対的データ汚染攻撃（Adversarial Poisoning Attack）

摂動を加えた汚染データを学習データに含め、これを標的AIに学習させることで、「トリガー」と呼ばれる攻撃者しか知り得ない入力パターンを、意図したクラスに誤分類させる。これを「バックドア攻撃」と呼ぶ。



Yansong Gao, et al., Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review, arXiv:2007.10760 (2020) から引用

バックドア攻撃

1. 通常、バックドア攻撃は「**標的型 (targeted)**」である。
2. 以下の２種類に分類される。
   - **クラス不可知論的 (Class-agnostic) 攻撃**
     トリガーの影響がソースクラスに依存しない場合
   - **クラス特異的 (class-specific) 攻撃**
     トリガーの影響がソースクラスに依存する場合
     （例）「STOP」の交通標識のみを「80マイル速度制限」に誤認させる。
3. 現時点での多くのバックドア攻撃の研究は前者。
4. トリガーになりうる情報は様々が提案されている。

Figure 4: Different means of constructing triggers. (a) An image blended with the Hello Kitty trigger [8]. (b) Distributed/spread trigger [83], [84]. (c) Accessory (eye-glass) as trigger [40]. (d) Facial characteristic as trigger: left with arched eyebrows; right with narrowed eyes [41].

(a) 背景のハローキティ
(b) パッチのばらつき
(c) アクセサリ（メガネ）
(d) 表情（左：アーチ状のまゆ、右：細目）

Yansong Gao, et al., Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review, arXiv:2007.10760 (2020) から引用

バックドア攻撃の種類と対策

攻撃対象		説明	対策
モデル汚染	コード汚染 (Code Poisoning)	AIフレームワーク上で構築されたMLモデルの脆弱性が残されたままになって、それを攻撃者に利用されて仕掛けられる。	•信頼できるソースコード提供者 •AIフレームワークやライブラリを最新版に更新 •AI稼働アカウントの権限制約 •サンドボックス環境でのテスト
	外部委託 (Outsourcing)	Machine Learning as a Service (MLaaS) などの外部業者にモデル構造や学習データを提供して、開発委託する際に仕掛けられる。	•信頼できるMLaaSや開発業者の選定
	事前学習モデル (Pretrained)	画像認識や文書解析などの大規模モデルの事前学習モデル（主に特徴抽出器）や教師モデルに仕掛けられる。	•信頼できる事前学習モデルや教師モデルを使用（不正にバックドア付きのものとすり替える可能性も考慮）
データ汚染	データ取得 (Data Collection)	信頼できないデータ提供元から汚染されたデータを取得し、学習することで仕掛けられる（データ汚染）。複数の提供元があるとき、信頼性を調査せずに取得してしまいがちで、現実的な脅威となっている。	•信頼できるデータ提供者を利用 •トリガー/汚染データの検知手法を適用
	協調学習 (Collaborative Learning)	連合学習(Federated Learning)や分割学習(Split Learning)では、データ所有者はデータを他の所有者と共有しなくてもAIモデルを学習できるため、攻撃者が混ざると、簡単にバックドアが仕掛けられる。	•信頼できる協力者と提携する。
	デプロイ後 (Post-deployment)	AIモデルがデプロイされた後、特に推論段階（ラベルが変更するため）で発生する。rowhammerバグを誘発することで、間接的に結合荷重のビットを反転させ、推論精度を低下させる。	•信頼できる計算基盤を利用する。

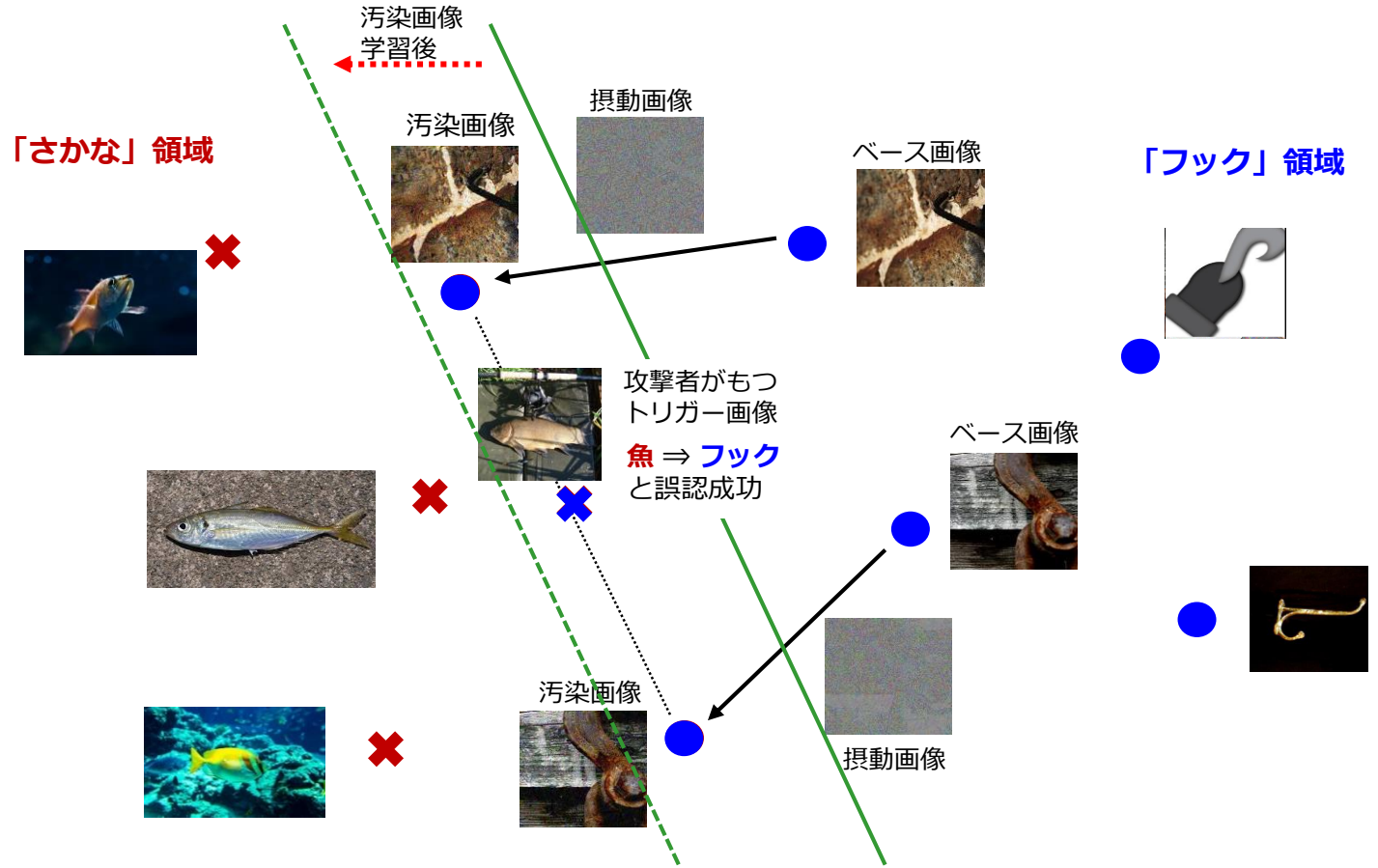
Yansong Gao, et al., Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review, arXiv:2007.10760 (2020) から引用

データ汚染・バックドア攻撃の方法

(a) 敵対的データ汚染攻撃（バックドア攻撃）

①Convex Polytope Attack

攻撃者は2つ以上の汚染データを、特定の入力パターン（トリガー）が誤判定するよう学習データに摂動画像を注入して、AIに学習させる。汚染データは見た目ベース画像と同じであるため、AI開発者が学習データ作成時に異常を検知することは困難

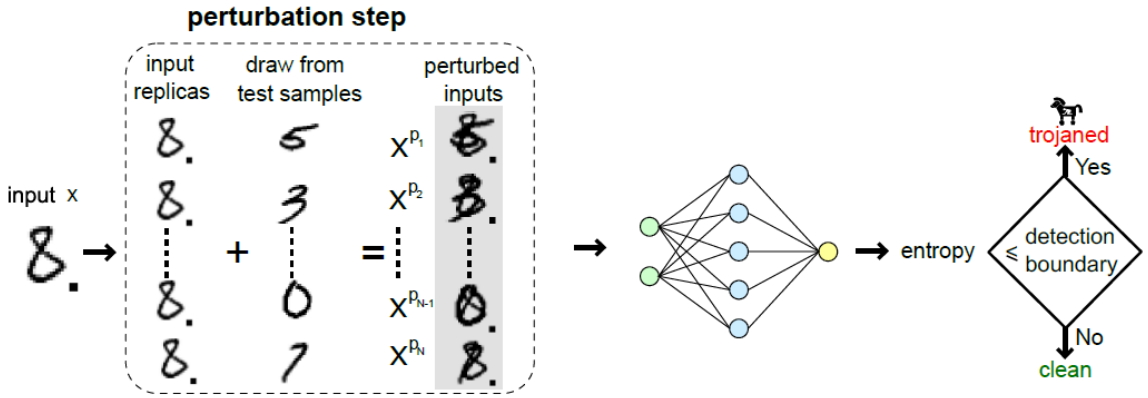


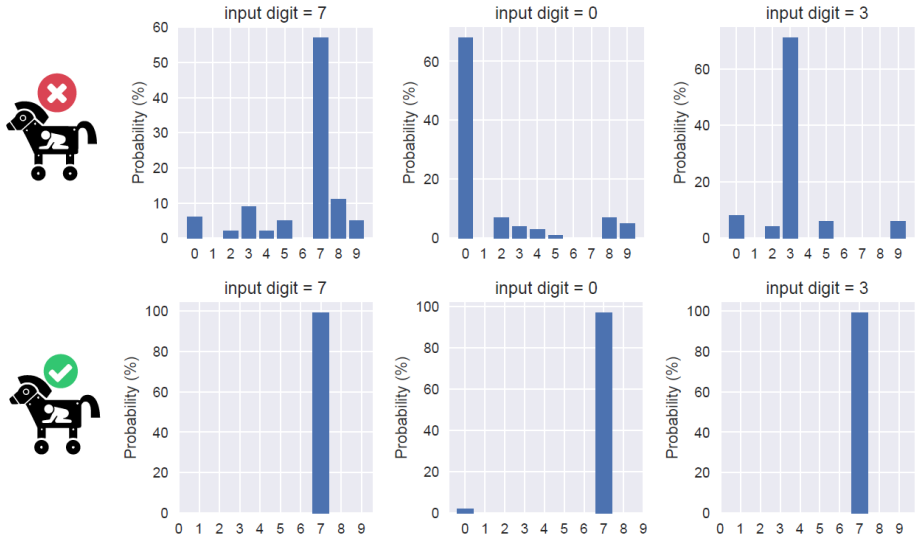
データ汚染・バックドア攻撃への対策

(a) トリガーの検知

①STRIP: STRong Intentional Perturbation

テスト画像を入力画像を重ねるなどの摂動を加え、ネットワーク出力のエントロピー（ばらつき）を観測し、トリガーを検知





エントロピー大： バックドアなし

エントロピー小： バックドアあり

➡ 入力に関わらず摂動を与えると「クラス７」となる確率が高い。これは「クラス７」と判定させる強い信号が内部に生成されていると考えられ、これがバックドアの根拠になる。

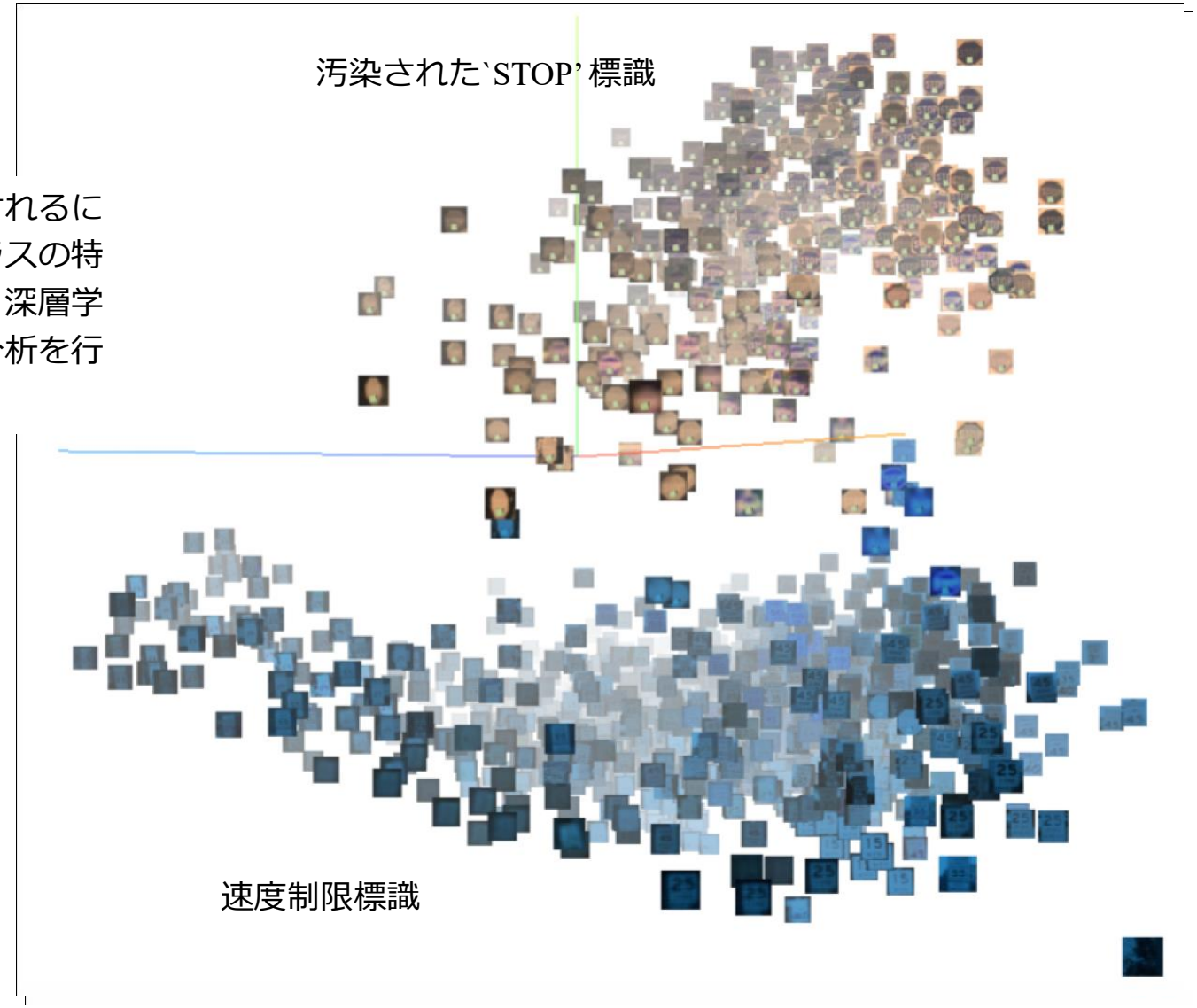
Y. Gao, et al., “STRIP: A defence against trojan attacks on deep neural networks,” ACM CCS 2019, pp. 113–125. から引用

データ汚染・バックドア攻撃への対策

(b) 汚染データの検知

①Activation Clustering

汚染データがターゲットクラスと判定されるには、汚染データの特徴がターゲットクラスの特徴と類似していることになる。そこで、深層学習モデルの最終層手前の出力を主成分分析を行って、クラスタリングする。



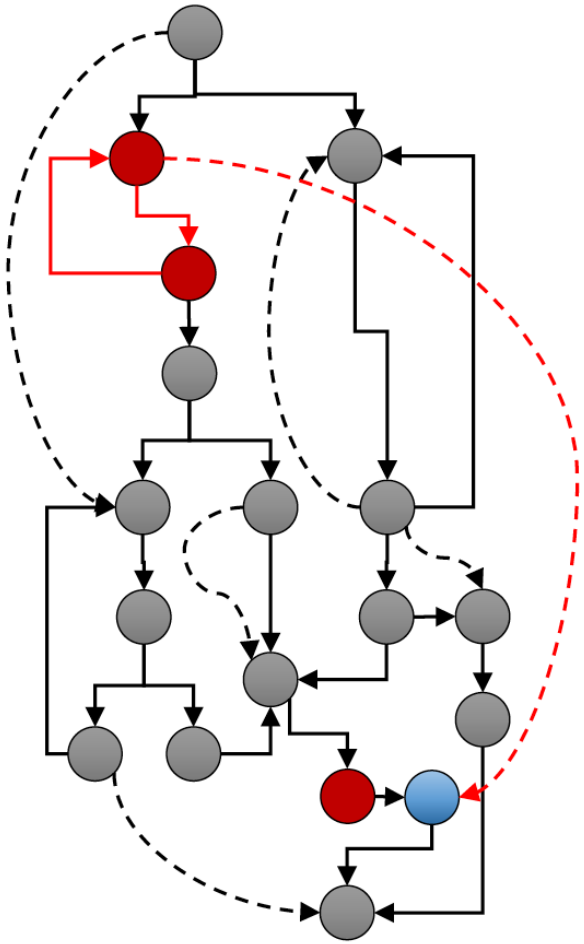
B. Chen, et al., “Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering,” arXiv:1811.03728 (2018) から引用

まとめ

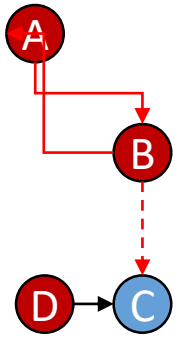
1. 多層ニューラルネットの万能性と脆弱性について述べ、人工知能研究のブレークスルーはAIモデルの情報資産としての価値を高める反面、サイバー脅威をもたらした。
2. AIもソフトウェアである。しかし、データ駆動型でシステム機能が実現される点が、従来のソフトウェアとは異なる。つまり、同じAIモデルのプログラムであっても、異なるデータで学習したものは違うAIモデルになり、機能も脆弱性も異なる場合がある。
3. AIの脆弱性は、モデルのソースコード汚染だけでなく、データ、開発環境（AIプラットフォーム、AIライブラリなど）、事前学習モデル、Dockerイメージ・コンテナなどの汚染でも引き起こされる。
4. AIへの攻撃は、AIの低脳化を狙った敵対的サンプル・回避攻撃、攻撃者にAIモデルを制御されるデータ汚染・バックドア攻撃、モデルや個人データの盗取、プライバシー侵害などがある。
5. 機械学習と用いたソフトウェアの脆弱性検知やバグフィックスなどの研究もある。

ご清聴ありがとうございました

Why graph? and How?



PDG example



Vulnerability part

Red: vulnerability trace.
Blue: potential manifestation point

Detection system

Remembering the pattern as a signature for certain vulnerability label either using rule-based or machine learning.

Why graph representation giving a good feature for source code vulnerability detection?

It is the simplest way to know what is going on in the machine that is written as high-level language (source code). It is more reliable than extracting the information from literal source code which can be obfuscated and minimized.

How?

By identifying the pattern of graph/subgraph which is related to the activity of the source code.
The pattern includes the structure and the node/edge feature information.
These information are relatively unique for every statement or expression

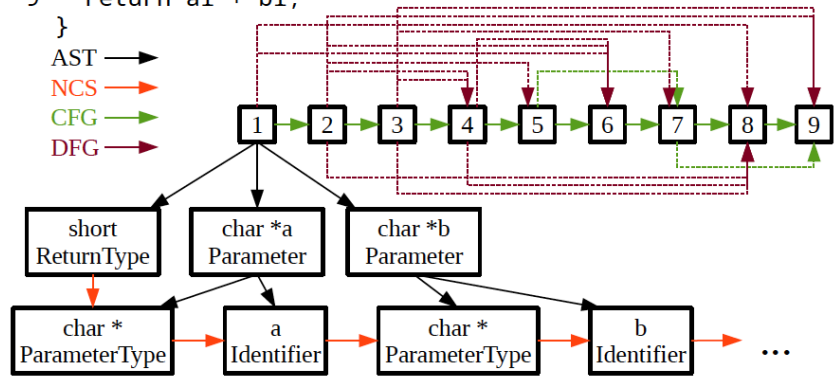
Limitations

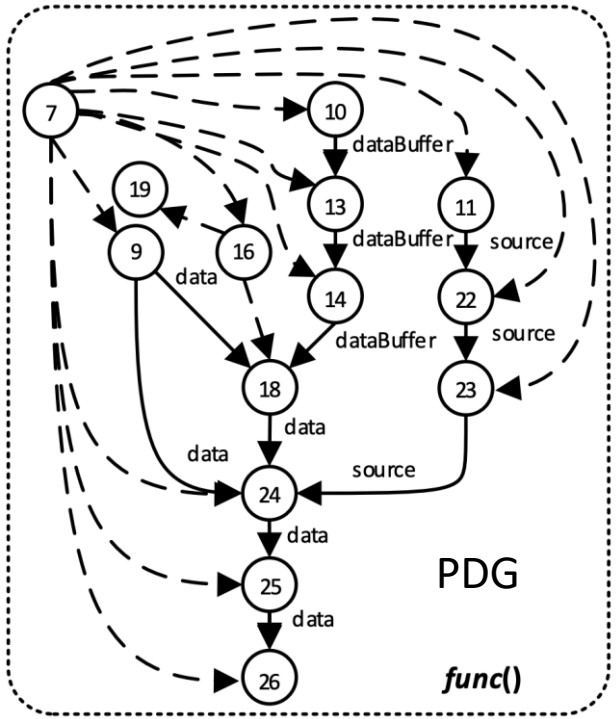
It is not real-time detection. The code must be ready for executed.
The dept and size of graph imply the expensive execution.

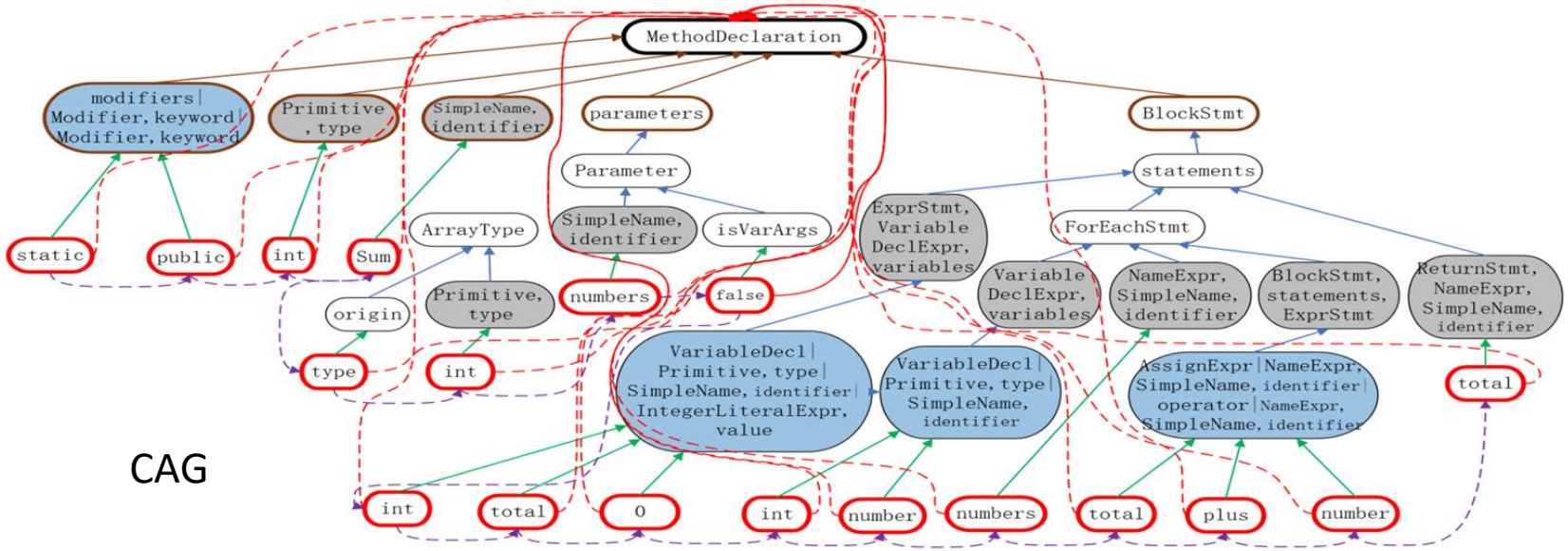
Source code graph representation

- Recently some papers used various graph representation such as
  - AST: abstract sytax tree
  - CFG: control flow graph
  - DFG: data flow graph
  - PDG: program dependency graph
  - CAG: compact abstract graph

```
1 short concat(char *a, char *b, char **out) {
2   short al = strlen(a);
3   short bl = strlen(b);
4   *out = (char *) malloc(al+bl);
5   if (al)
6     memcpy(*out, a, al);
7   if (bl)
8     memcpy(*out+al,b,bl);
9   return al + bl;
  }
```







CAG

Hoppity: Learning to Fix Bug

- Elizabeth Dinella et al., “Hoppity: Learning Graph Transformations To Detect And Fix Bugs In Programs.” ICLR’2020