機械学習における品質保証のチャレンジ

JEITAソフトウェアエンジニアリング技術ワークショップ2017

国立情報学研究所　石川　冬樹

f-ishikawa@nii.ac.jp / @fyufyu

http://research.nii.ac.jp/~f-ishikawa/

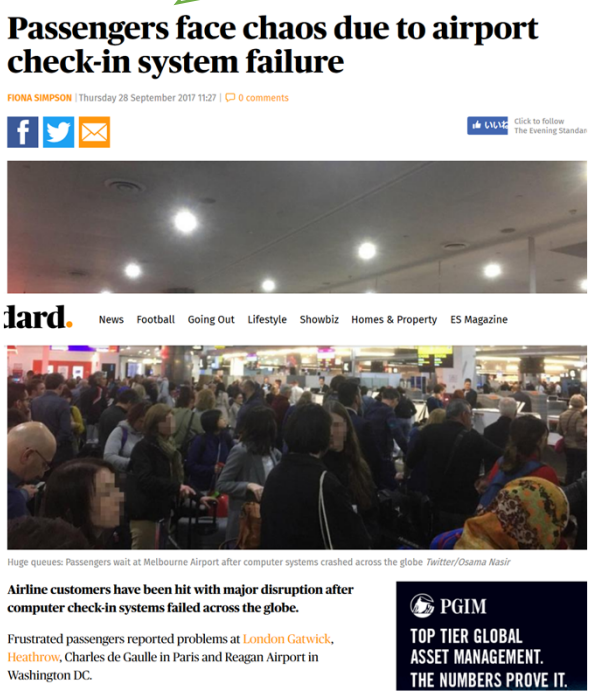
大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

イメージをもつための事例

はじめに

- ソフトウェアシステムの「品質」
  - 残念ながら個人・組織・社会に損害をもたらす問題も多々起きてしまっている
  - とはいえ技術的な進歩・累積も大きい

一例：2017/09/28



機械学習あるいはAIを用いたシステムは何が違うのか？？

[https://www.standard.co.uk/news/world/passengers-face-chaos-after-airport-checkin-systems-crash-across-globe-a3645816.html]

社会的影響の大きな例

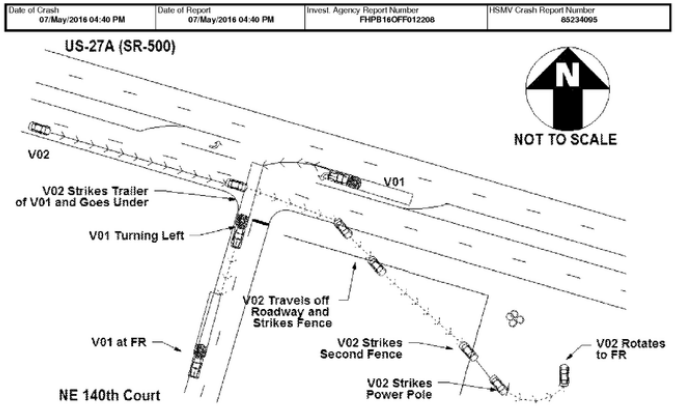
- 2016年のテスラ自動運転車の事故　*ここが機械学習*
  - テスラの発表：「まぶしい空に対してトレーラーの白い側面を認識できず」（「運転者も」ともある）

[https://www.tesla.com/jp/blog/tragic-loss]

  - 注：2017年11月の調査報告での論点に上記は全くなく，「そもそも自動運転の対象外状況だが運転者が長らく操作していない」という過信や，アラートの効力が中心

[https://dms.ntsb.gov/pubdms/search/hitlist.cfm?docketID=59989]

[http://www.dailymail.co.uk/news/article-3677101/Tesla-told-regulators-fatal-Autopilot-crash-nine-days-happened.html]



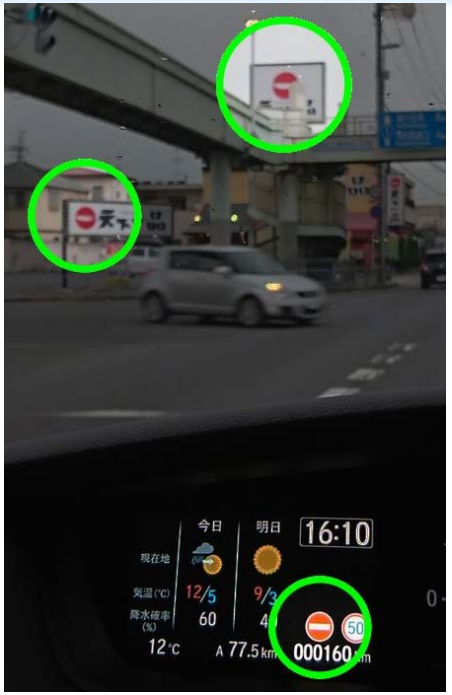
関連した例（１）

- 最近のTweetより（Honda SENSING）

[https://twitter.com/_gyochan_/status/938240168078622720]
[https://twitter.com/Bleu_kakeru727/status/937680760491753473]

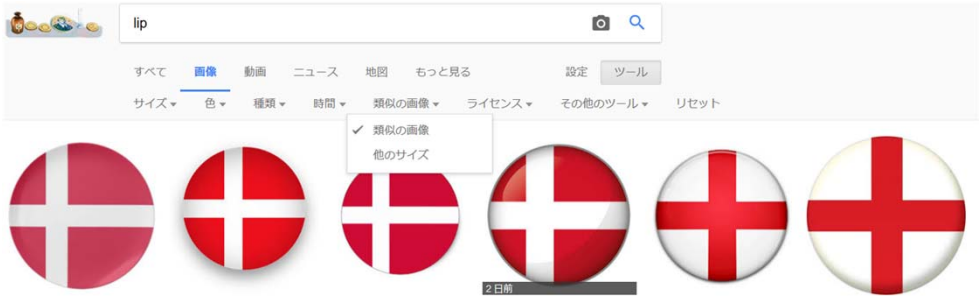
[http://www.tenkaippin.co.jp/company.html]

注：利用者への注意あり
http://www.honda.co.jp/hondasensing/feature/srf/



- おまけ： Google画像類似検索
  - ロゴイラスト部のみ切り取って検索

➡「たぶん”lip”」といいつつ，デンマーク国旗と（その赤白逆の）イングランド国旗をまず推してくる



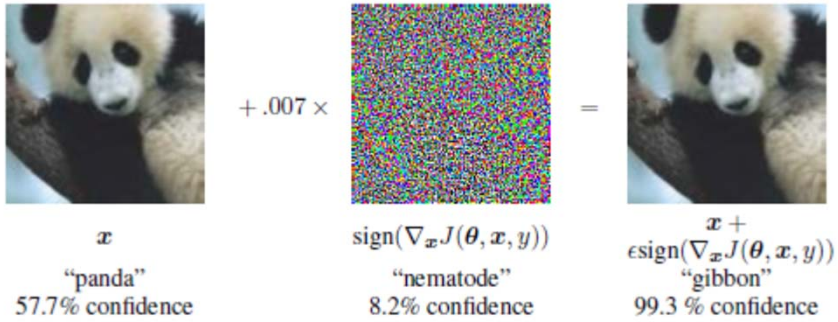
[https://www.google.co.jp/imghp] (2017/12/11)

（進入禁止標識は出ず）

（←これは40位くらい）

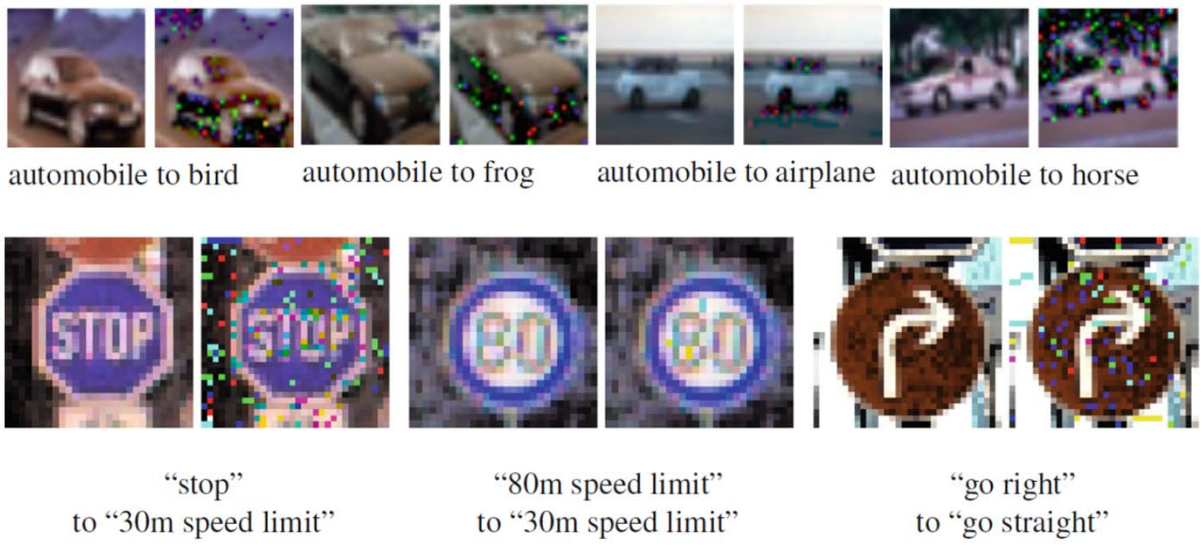
関連した例（２）

- Deep Learningに関する論文より



有名な例
「パンダ」が「テナガザル」に

[Goodfellow et al., Explaining and Harnessing Adversarial Examples, 2015]



[Huang et al., Safety Verification of Deep Neural Networks, 2017]

ちょっと違う例

- Twitter Botによる（ユーザを真似た結果の）不適切発言

If you guessed, “It will probably become really racist,” you’ve clearly spent time on the Internet. Less than 24 hours after the bot, @TayandYou, went online Wednesday, Microsoft halted posting from the account and deleted several of its most obscene statements.

The bot, developed by Microsoft’s technology and research and Bing teams, got major assistance in being offensive from users who egged it on. It disputed the existence of the Holocaust, referred to women and minorities with unpublishable words and advocated genocide. Several of the tweets were sent after users commanded the bot to repeat their own statements, and the bot dutifully obliged.

Tay’s Twitter account. The bot was developed by Microsoft’s technology and research and Bing teams.

[https://www.nytimes.com/2016/03/25/technology/microsoft-created-a-twitter-bot-to-learn-from-users-it-quickly-became-a-racist-jerk.html]

要求と不具合・運用について考える例

[https://twitter.com/RapidUTL/status/932189705088598017/photo/1]
(2017/11/19)

「あら」の扱いが修正？？

“Fish Soup”になるわけではない？

[https://translate.google.co.jp/]
(2017/12/11)

以下で同時期（11月中旬）に話題になったような「他の変なもの」も挙動が変わっている？
https://togetter.com/li/1173236

例題チョイスで遊びすぎました

- でも「似たような」技術で，皆さん
  ビジネスや実世界の制御やるのですよね？
  - 物体認識に基づく自動運転
  - 音声認識に基づくAIスピーカー・AI家電など
  - 商品の売上げ予測に基づく生産・発注制御など
  - バグ予測や工数予測に基づくプロジェクト管理判断など
  - ・・・

（ロゴの類似画像検索や「あら汁」などは
影響が致命的なユースケースはまぁないかもしれないが）

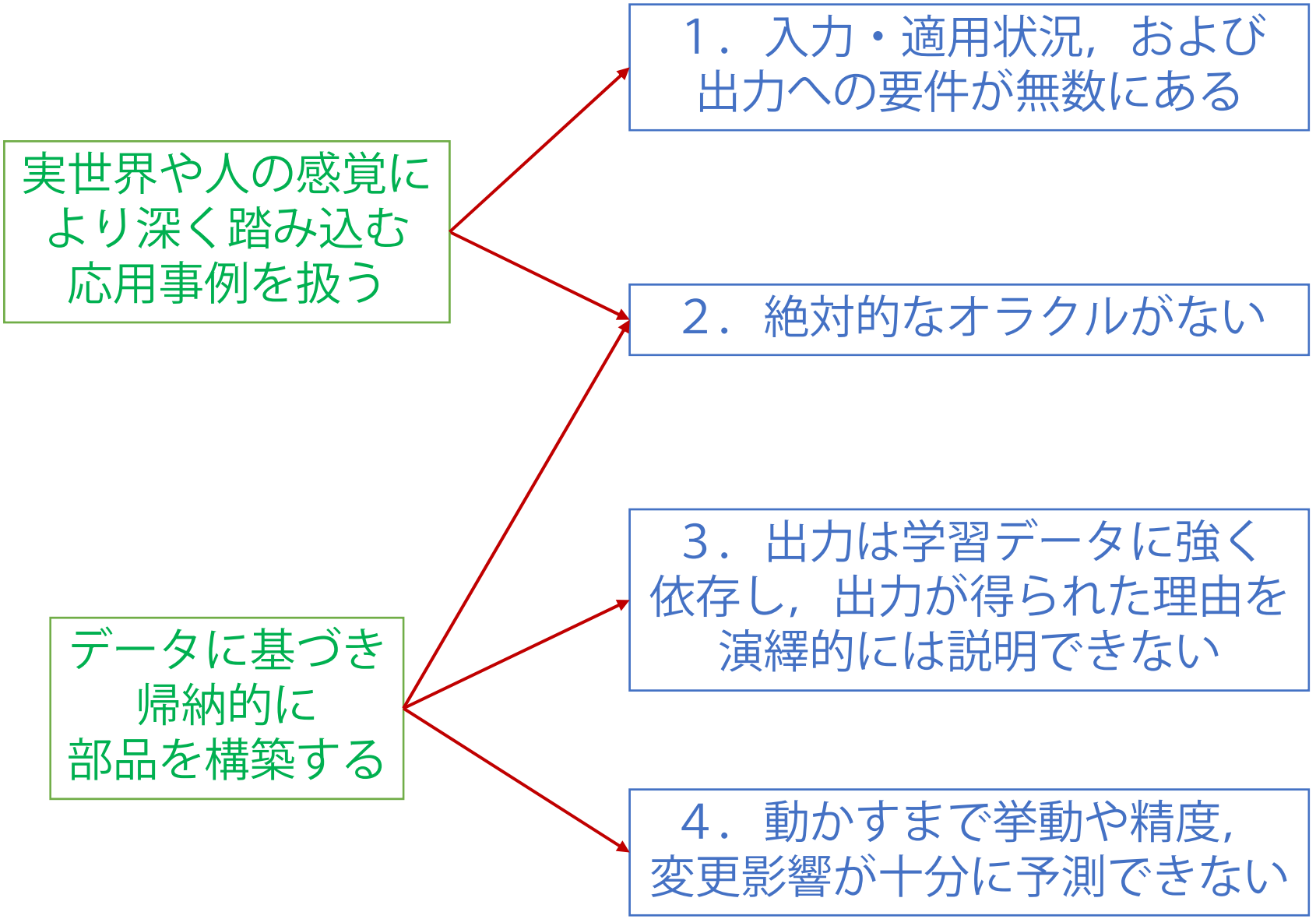
皆さんのシステムで起きうる「品質問題」の影響は？

何を考えますか？

- これらすべて「バグ」か？潰さなければならないのか？
- そもそも潰せるものなのか？それとも技術的限界？
- どうやってこれらに気づくのか？他に対処すべき「同種」や「類似」の状況は列挙できるのか？どうやって？
- そもそも何を持って「品質保証」「完成」とするのか？「仕様」は何なのか？事前に見積・合意できるのか？
- 修正内容をどう決めてどれだけの頻度で更新するのか？
- 顧客やユーザには何をどうやって説明するのか？
- こんな挙動の原因を把握したり，予測し踏まえてテストしたりできるのか？どうやって？開発者ならわかるものなのか？（外部）品質保証者は？
- ・・・

根本的なスタンスの変化

本質的な難しさの要因（の一部）（１）



本質的な難しさの要因（の一部）（２）

1．入力・適用状況，および出力への要件が無数にある

2．絶対的なオラクルがない

3．出力は学習データに強く依存し，出力が得られた理由を演繹的には説明できない

4．動かすまで挙動や精度，変更影響が十分に予測できない

実行時に想定外のことが発生することが原則になる

テスト入力を多数用意しても成否判定が困難・高コストで，明らかなコーディングミス等すら気づかない可能性がある

何が実際実現できているのか把握・説明できず，問題の原因同定・修正も難しい

要求の実現可否・実現コストや変更の影響を事前に把握しての分析・意思決定は難しい

本質的な難しさの要因（の一部）（３）

実行時に想定外のことが
発生することが原則になる

テスト入力を多数用意しても
成否判定が困難・高コストで，
明らかなコーディングミス等
すら気づかない可能性がある

何が実際実現できているのか
把握・説明できず，問題の
原因同定・修正も難しい

要求の実現可否・実現コストや
変更の影響を事前に把握しての
分析・意思決定は難しい

ん？全部「不確かさ」の
話じゃないか！

要求と評価指標，
入力，実装，出力が不確か

自然に必要となるだろうスタンス

実行時に想定外のことが
発生することが原則になる

より上位のシステム全体で
失敗の対応策を打つ

テスト入力を多数用意しても
成否判定が困難・高コストで，
明らかなコーディングミス等
でも気づかない可能性がある

実行時に継続的な監視・評価を
行いフィードバックする

システムの上位の目的
（ユーザ体験や安全性）の観点
からフィードバックを得る

何が実際実現できているのか
把握・説明できず，問題の
原因同定・修正も難しい

疑似オラクルを用意し
大量に（ゆえに自動）テストをする

構築過程から部分部分に対し
確信が持てるような構築方法や
検証手段を設ける
（想定する性質のアサーション等）

要求の実現可否・実現コストや
変更の影響を事前に把握しての
分析・意思決定は難しい

探索的・試験的に
仮説・分析・評価を繰り返す

不確かさへのアプローチといえば

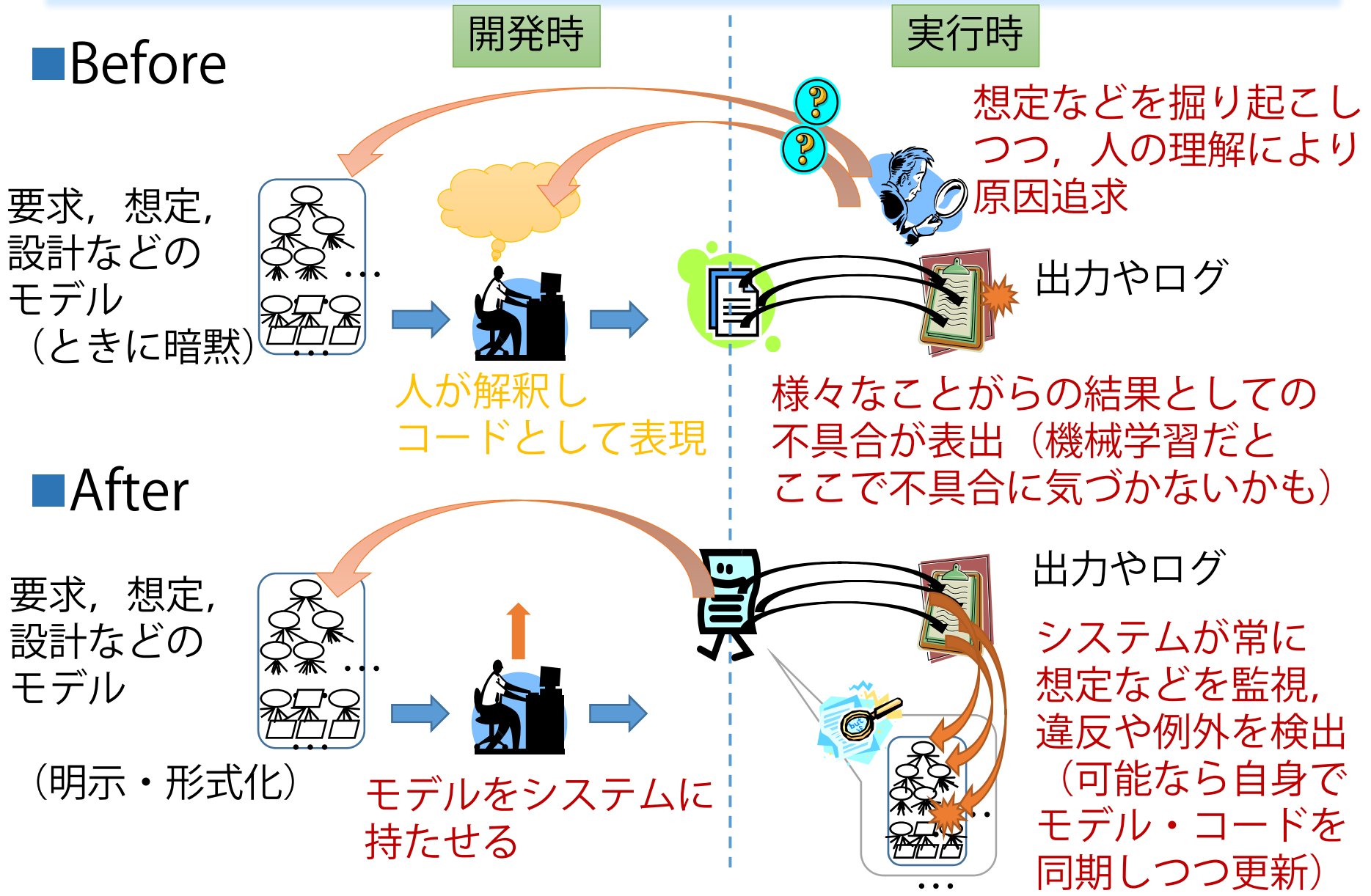
- Models@run.timeアプローチとの共通点大

要求，仮定・想定や，設計に関する意思決定の論理などのモデルを明示化し，システムが実行時にも活用

➡ *現実とモデルのずれを検出・同定しやすく，また意思決定の論理をたどり直して自動設計変更などのフィードバックも可能に*

- CPSや自己適応に関連するソフトウェア工学研究において2010年代盛んに取り組まれている一つの方向性
- ただし帰納的な振る舞いを主眼においてはこなかった

補足： Model@run.time



補足： Model@run.time

- 様々な言葉
  - “Models need to continue to live at run-time and evolve as changes occur while the software is running”

    [Baresi et al., The Disappearing Boundary Between Development-time and Run-time, 2010]

  - “Run-time reasoning over requirements is necessary where design-time decisions about the requirements are made on incomplete and uncertain knowledge about the domain and the stakeholders’ goals”

    [Sawyer et al., Requirements-Aware Systems - A research agenda for RE for self-adaptive systems, 2010]

  - “a runtime model can be seen as a live development model”

    [Blair et al., Models@Run.Time, 2009]

参考になるアプローチ？？（1）

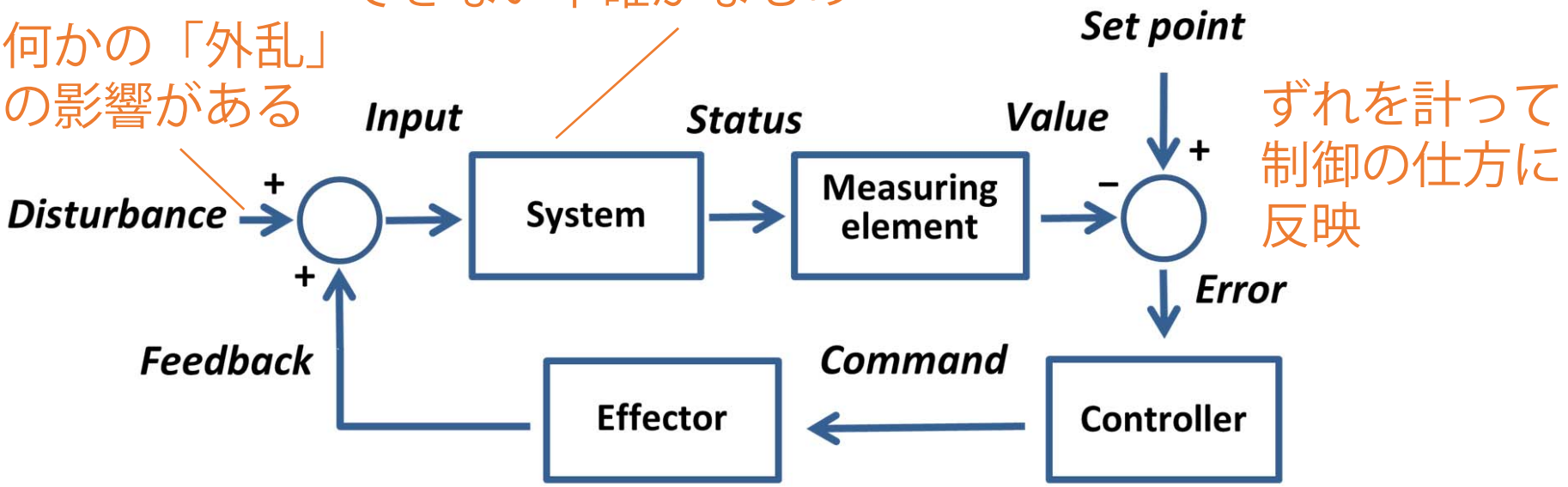
（V字モデルなど眺めるよりむしろ・・・）

- 制御工学におけるフィードバックループ
  - 「自己適応システムのためのソフトウェア工学」の人たちには常識 [Filieri, Control theory for software engineering: technical briefing, 2016]

制御したい対象：思い通りにできない不確かなもの

何かの「外乱」の影響がある

ずれを計って制御の仕方に反映

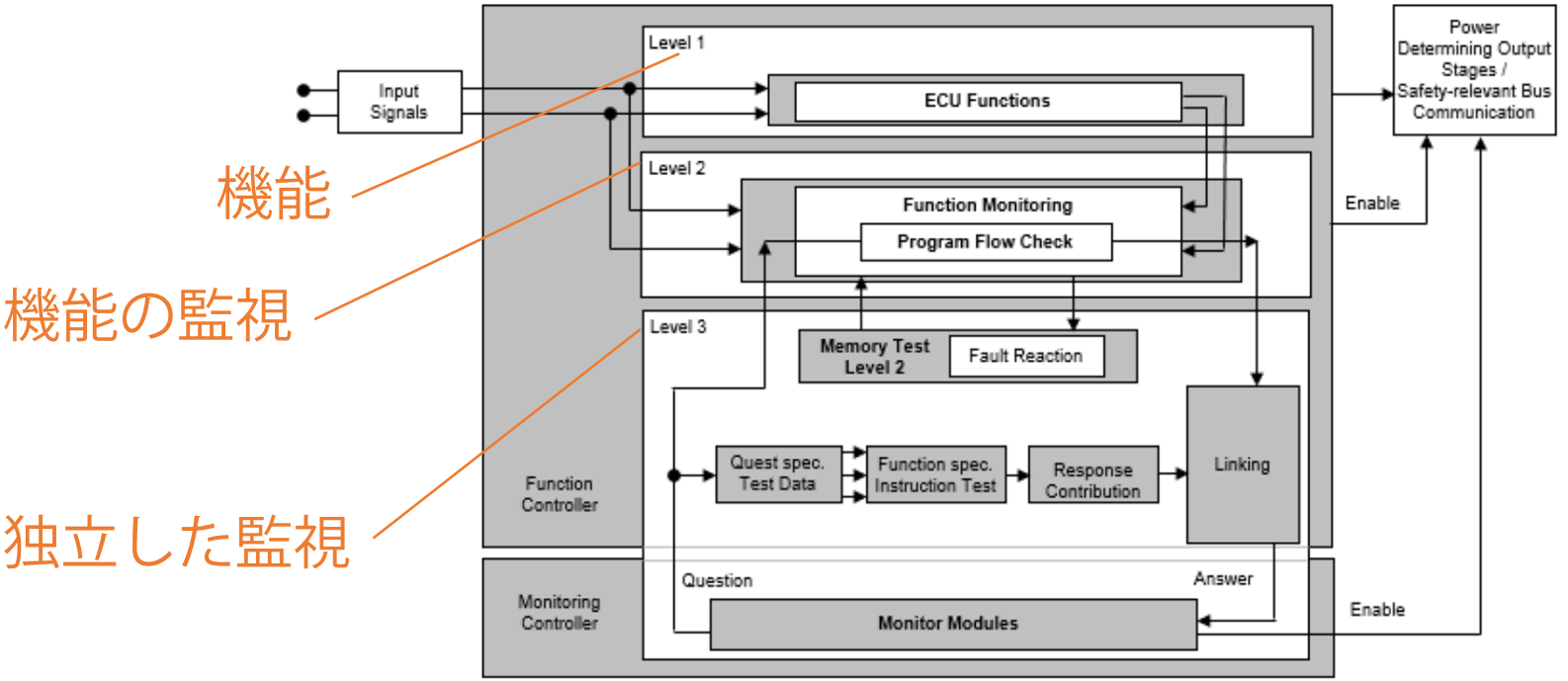


[https://en.wikipedia.org/wiki/Feedback]

参考になるアプローチ？？（2）

（V字モデルなど眺めるよりむしろ・・・）

- 例：車におけるエンジン周り監視の「3レベル」
  - アウディ，ダイムラー，ポルシェ，フォルクスワーゲンで「差別化要素がない」として標準化



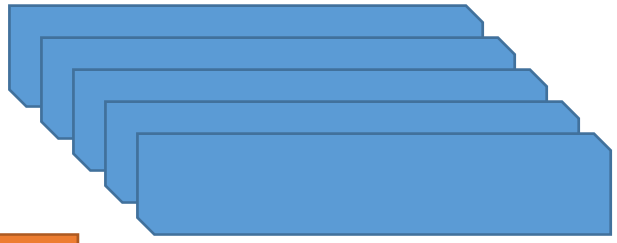
[Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units Ver. 6.0, 2015]

品質保証のための
原則？？パターン？？

ものすごく概念的には・・・

対象ML部品

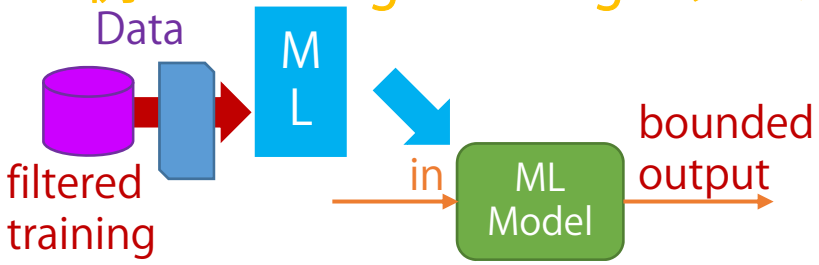
対象MLシステムに対する
要件や疑似オラクル（比較対象など）



個々の要件・疑似オラクルに対する
- 実装手段：演繹的 or 帰納的,
  Correctness-by-Construction or 検証・テスト
- 反映アーキテクチャと
  反映タイミング（オフライン・オンライン）

例：Output Boundパターン

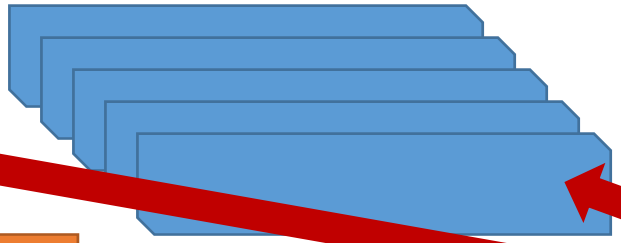
例：Training Filteringパターン



ものすごく概念的には・・・

対象ML
部品

対象MLシステムに対する
要件や疑似オラクル（比較対象など）

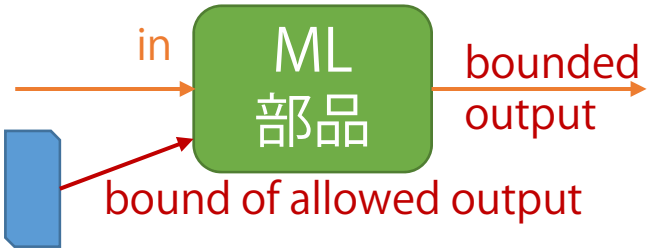


監視

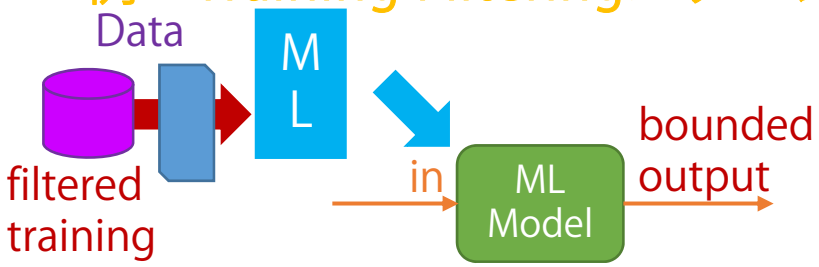
フィードバック

個々の要件・疑似オラクルに対する
- 実装手段：演繹的 or 帰納的,
  Correctness-by-Construction or 検証・テスト
- 反映アーキテクチャと
  反映タイミング（オフライン・オンライン）

例：Output Boundパターン



例：Training Filteringパターン

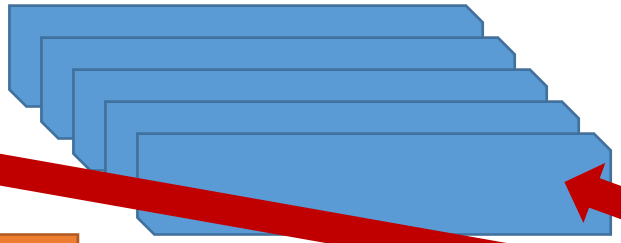


ものすごく概念的には・・・

これが一つの部品に

対象ML
部品

対象MLシステムに対する
要件や疑似オラクル（比較対象など）

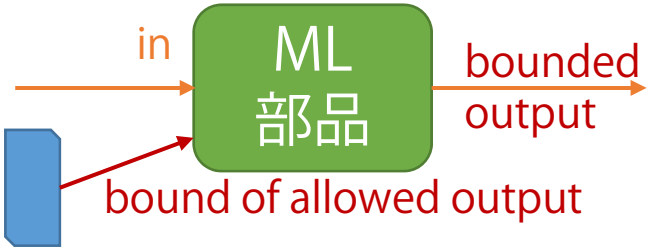


監視

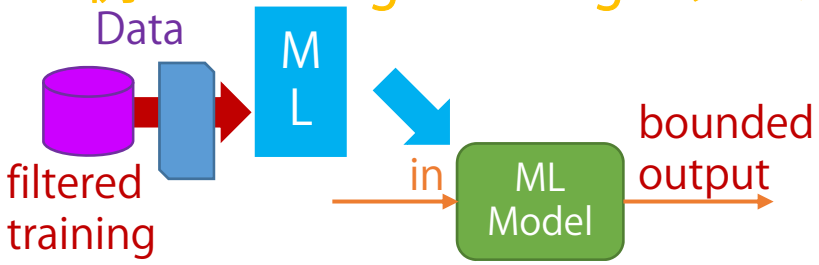
フィードバック

個々の要件・疑似オラクルに対する
- 実装手段：演繹的 or 帰納的,
  Correctness-by-Construction or 検証・テスト
- 反映アーキテクチャと
  反映タイミング（オフライン・オンライン）

例：Output Boundパターン



例：Training Filteringパターン



・・・みたいな共通原則やパターンなど

- 何も（広くは）確立されていない
  - Googleなどからは少しずつ出てきている
  - 先の実装パターン2つは適当な例（ちょっと雑）
  - 全体像は先週モデルベース開発系の研究者たちと議論した結果出てきたもの（の石川個人の解釈・思想）
  - テスト技術など個別の研究はちらほらという感じ
- 現状多くの企業では，何もかもが探索的・試験的にならざるを得ない
- だからこそ，今の常識・プロセスにとらわれず，あるべき姿を追求するチャンスでもある？？

具体的な技術

補足：先人のヒント

- NIPS，ICML等の機械学習系国際会議
  - 特に併設ワークショップの”Reliable Machine Learning in the Wild”や”ML Systems”など，アルゴリズムよりも実用システム構築上の課題に着眼したもの
  - Googleなど企業の招待講演が多く参考になる
- ソフトウェア工学系国際会議ではまだまだ
  - 間接的にヒントになりそうなものは多い：継続的適応・進化のとらえ方，疑似オラクルを用いたテスト手法，確率的な出力があるソフトウェアのテスト手法などなど
  - 特にサーチベースドテスティング，メタモルフィックテスティング

補足：

- メタモルフィックテスティング
  - 「入力を変えると出力はこう変わるはず」という関係を活用し既存テストケースから多数のテストケースを生成
  - 例：一部データを除いても，残りのもののランキングは変わらない

[Segura et al., A Survey on Metamorphic Testing, 2016]
[中島,データセット多様性のソフトウェア・テスティング, 2017]
[Ding et al., Validating a Deep Learning Framework by Metamorphic Testing, 2017]

- サーチベースドテスティング
  - メタヒューリスティックを用いてスコアを最大化するようテストスイートやテストケースを生成

[http://www.evosuite.org/]

参考（１）：原則・指針

- 機械学習における「技術的負債」
  - 従来のソフトウェアにおけるものに加えて，固有の負債が多数ある難しさ
  - 例： Changing Anything Changes Everything

[Sculley et al., Machine Learning: The High-Interest Credit Card of Technical Debt, 2014]

タイトルに注目！

- ベストプラクティス集
  - 例： watch for silent failures

[Zinkevich, Rules of Machine Learning: Best Practices for ML Engineering, 2016]

- どれだけ監視・テストできているかの評価スコア
  - 例： 個々のfeatureの分布が予想と合っているかどうか

[Breck, What's your ML Test Score? A rubric for ML production systems, 2016]

参考（２－１）：検証・テスト

- ディープラーニングのホワイトボックステスト
  - サーチベースドテスティング
  - ニューロンのカバレッジを上げるとともに，比較対象の別実装との出力差分を上げるように，テスト入力セットを最適化

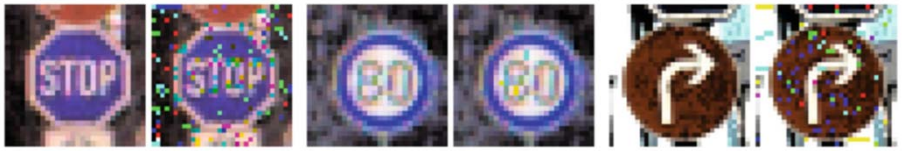
[Pei et al., DeepXplore: Automated Whitebox Testing of Deep Learning Systems, 2017]

- GAN（Generative Adversarial Networks）
  - 機械学習コミュニティでは盛ん（ここでは軽く）
  - 雑には，品質保証・向上対象の機械学習部品に対して，「敵対的」な機械学習部品をぶつけ，同時に育てる（「失敗のさせ方」を学ぶということ）

[Goodfellow et al., Generative Adversarial Networks, 2014]

参考（２－２）：検証・テスト

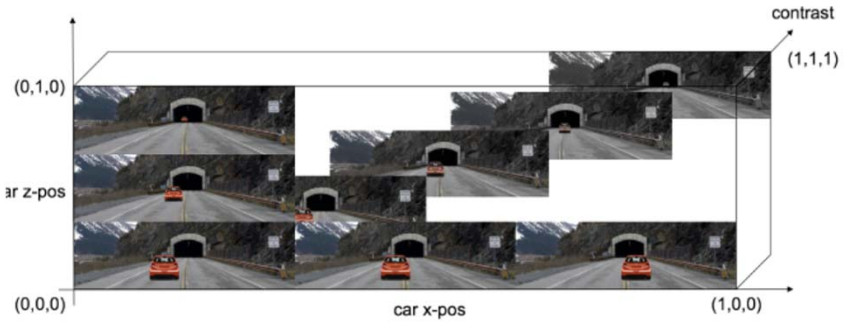
- 形式検証技術の応用
  - 画像認識において，「一定の操作」を行っても認識結果が変わらないかどうかを網羅的に検証



"stop" to "30m speed limit"　　"80m speed limit" to "30m speed limit"　　"go right" to "go straight"

[Huang et al., Safety Verification of Deep Neural Networks, 2017]

- 系統的な画像生成によるテスト
  - 認識できない「穴」が見つかった



[Drossi et al., Systematic Testing of Convolutional Neural Networks for Autonomous Driving, 2017]

おわりに

※ ここまでの内容はあくまでごく一側面

今までソフトウェア屋さんがサボってきた・やりきれてなかったことが突きつけられているのかも

- 画面の外の実世界での意義や安全性，ユーザの感性的な満足の追求や評価，探索的・試験的な投資や問題解決，仮説と検証による科学的な判断や評価，常に測定，数学の理解・活用，Correctness-by-Construction，現実問題の数学的問題（最適化等）への帰着，実行時検証やテスト自動化（入力生成・疑似オラクル），といったことへの本気のパラダイムシフト，・・・

楽しんで切り拓いていきましょう！

新研究会などもやっていくのでよろしくお願いします！