組込みの生成AIの可能性

JASA北海道交流セミナー2023

中村 仁昭

自己紹介

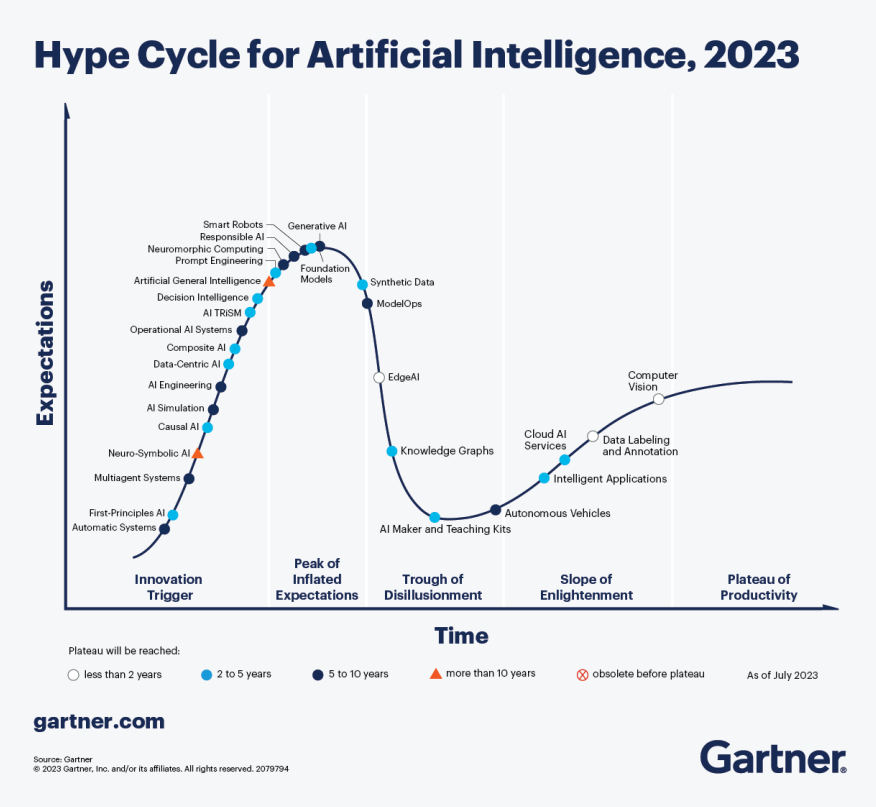
- 株式会社Bee CTO、JASA AI研究WG主査 中村 仁昭
  - 社内AIチームPM、車載プラットフォームPMを担当
  - 2018年からJASAでAIセミナー、研究会を実施
- CQ出版 Interface誌でAI含めた様々な記事を執筆
  - 2023年10月号から「Rustプログラミング問題集」の連載開始

アウトライン

- 組込みAIの現状
- 組込みの生成AIの可能性
- JASA AI研究WGの紹介

組込みAIの現状

組込みAI(EdgeAI)の位置付け



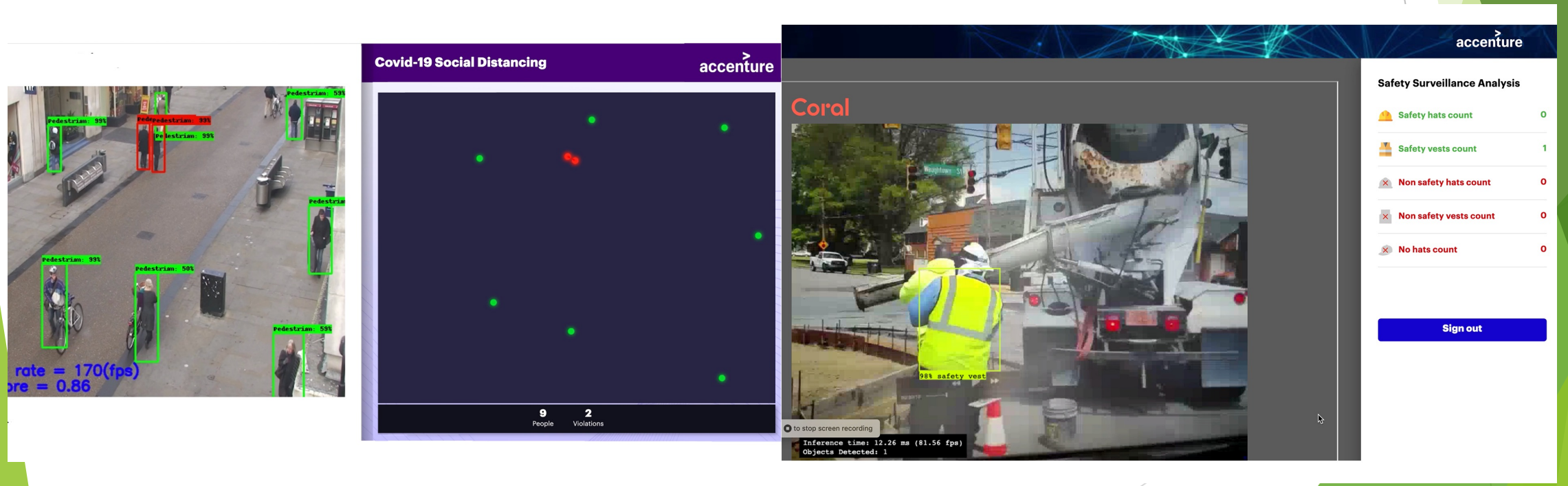
- GartnerのHype Cycle for Artificial Intelligence，2023ではようやく幻滅期に入り、ますます実装例は手に入りにくい状況に
  - 主流の採用までに要する年数は2年未満とされている

組込みAIの実装例

- スマホの音声認識や、カメラ画像処理は採用されて久しい
- Intel CoralのCustomer Storiesが数少ないながらも実際に使用されている実例が更新されている
  - コンサルティング会社AccentureのAIを活用した外観検査の取り組み
  - くら寿司の皿を数える装置
  - ノルウェーの配電会社Pratexoの電力グリッドの変圧器の異常検査
  - など

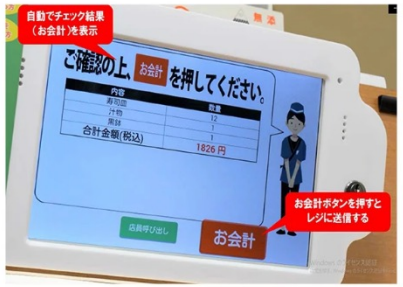
Accentureの外観検査の取り組み

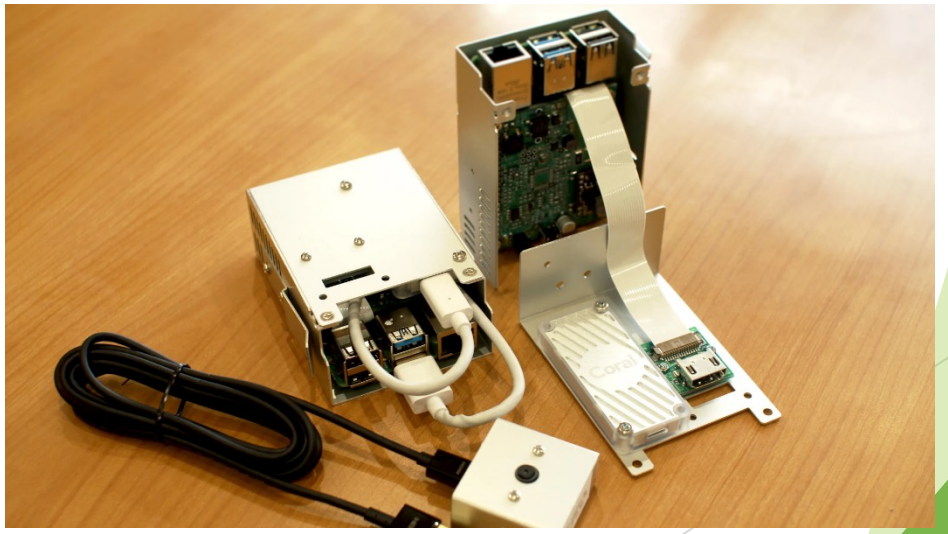
- 知的財産と労働者を保護するためにプライバシーとセキュリティを強化する必要があり、ネットワークを使用しないローカルの分析にCoral EdgeAIを選択



くら寿司の皿を数える装置

- 回転レーンから取った皿の数をRaspberry　Pi4でQRコードの識別とTensorFlow(Coral　USB　Accelarator)を使った画像認識で皿の種類と数をカウント





Pratexoの電力グリッドの変圧器の異常検知

- Coral M.2アクセラレーターで各変圧器が発する音から機械学習モデルで問題が発生するかを予測し電力グリッドの信頼性を確保

組込みAI実装の普及に向けて

- 実装例が表に出てこない
  - 調査は継続する必要はありそう
- もう幻滅期に入ってメディアに注目されないため、自ら実装を進めないと情報が集まらない可能性が高い
  - Computer Visionなど啓発期に入っている技術を組込みに導入するのであれば、低リスクで実装を進められる

組込みの生成AIの可能性

組込み環境に生成AIを実装してみる

- どのような生成AIを実装するか？
- 様々なモデルが存在するが生成AIブームのきっかけは
  - Midjourneyなどの画像生成モデル
  - ChatGPTなどの大規模言語モデル(LLM: Large Language Model)
- 特定領域でなく様々な応用が見込めるLLMを実装してみたい
- LLMで公開されているLlama2を実装してみる

Llama2

- 2023/7/18にMeta(Facebook)が公開したLLM
- 先に開発されたLlamaの最新版で商用利用も可能なモデル
  - 公開モデルとしては性能が高い
  - OpenAIのGPT-4などのクローズドなLLMと競合する形で、オープンモデルのデファクトスタンダードになりつつある
- サイズは70億、130億、700億の3種類
- Llama2をベースに日本語による追加事前学習を行なった日本語言語モデルELYZAがELYZA, Inc.により公開されているのでこれを採用
  - ELYZA, Inc.は東京大学 松尾研究所発のAIカンパニー
  - ELYZAもLlama2同様に商用利用可能なモデルとして公開

動作環境

- 入手が容易なSBC(Single-board computer): Raspberry Pi 4
- AI性能が高いSBC: Jetson Xavier NX
- Llama2/ELYZAはHugging Face Hubで公開されていて、Pythonのtransformersライブラリから利用可能
  - 依存するライブラリも多く環境構築に時間がかかる
  - なるべく簡単に環境を構築したい
- Llama系モデルの推論をC/C++で実行できるllama.cppを採用
  - 依存なしの純粋なC/C++実装なためビルドしてすぐに実行可能
  - ARM NEONやx86 AVX、NVIDIA CUDAなど様々な最適化に対応
  - 2bit～6bit、8bitの量子化をサポート

Raspberry Pi 4

- Raspberry Pi OS (64-bit): 2023-05-03-raspios-bullseye-arm64.img.xz
  - 32-bit(2023-05-03)版はカーネルのみ64bit(aarch64)でビルドが失敗…
- 実行時
  - メモリ消費: 3.9GB
  - CPU使用率: 4コア全て100%
- NEONのみ有効な状態で推論が0.84 tokens/sec

Raspberry Pi 4

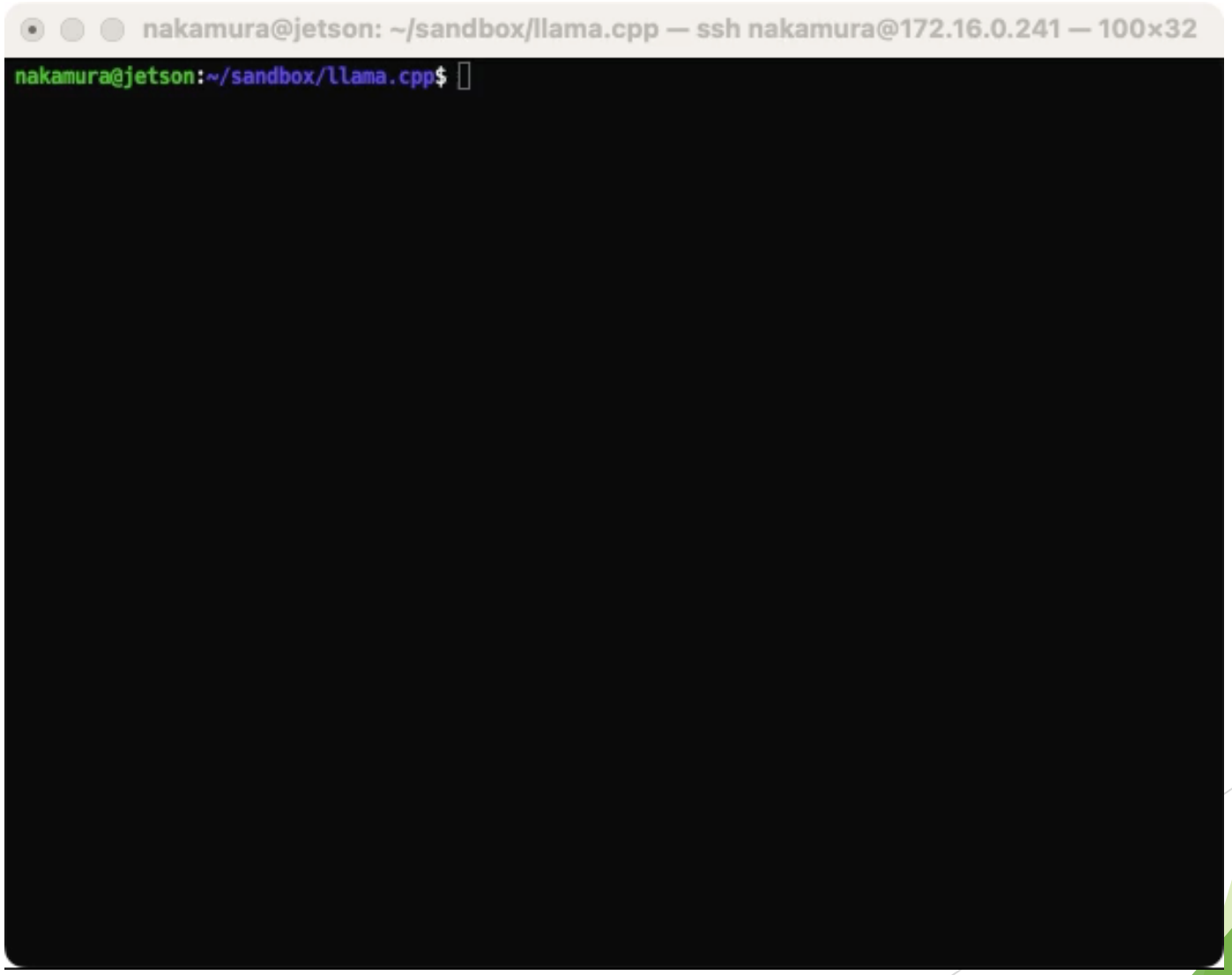
Jetson Xavier NX

- JetPack 5.1.2
- CUDA(cuBLAS)有効化ビルド(make LLAMA_CUBLAS=1)はそのままではエラー
  - インストール直後の環境ではnvccコンパイラにパスが通っていない
  - nvccのarchオプションがnativeではエラーになるため、Makefileを編集

```
$ git diff
diff --git a/Makefile b/Makefile
index a774dc5..f6435be 100644
--- a/Makefile
+++ b/Makefile
@@ -338,7 +338,7 @@ endif #LLAMA_CUDA_NVCC
 ifdef CUDA_DOCKER_ARCH
        NVCCFLAGS += -Wno-deprecated-gpu-targets -arch=$(CUDA_DOCKER_ARCH)
 else
-       NVCCFLAGS += -arch=native
+       NVCCFLAGS += -arch=compute_72
 endif # CUDA_DOCKER_ARCH
 ifdef LLAMA_CUDA_FORCE_DMMV
        NVCCFLAGS += -DGGML_CUDA_FORCE_DMMV
```

- CUDA有効(20W設定)で推論が10.97 tokens/sec
- 推論時 メモリ消費: 2GB、CPU使用率: 1コア(100%)、 2コア(数%)

Jetson Xavier NX



動作速度

model	Tokens/sec
Raspberry Pi 4	0.84
Jetson Xavier NX	10.97
GeForce RTX 4070Ti(12GB) *1	12.01

*1 Windows環境でLlama2を動作させた際の速度(リンク)

- Raspberry Pi 4はさすがに遅い
  - 音読の速度よりも遅いため音声読み上げも厳しい
- Jetson Xavier NXは快適に動作
  - ChatGPTより速い印象
  - デスクトップPCとほぼ同レベルの速度

組込み生成AIの実用性

- テキストベースのコンソールでのやり取りはGPUがあれば十分実用的
- テキストを扱うユースケースが組込みでは少ない？
  - EPG(電子番組表)や歌詞など外部から入力されるテキストを扱う場合は様々な用途がありそう
  - 入出力がそもそも少ない機器はローカルに膨大な知識を持ったとしても、他の手段の方がコスパが当分の間良さそう
    - 日照時間や温度から緯度の推定？ 素直にGPSやWifiが正確で安価...
  - 内部ログから異常検出ぐらい？
- 入出力が音声/映像が可能なマルチモーダルなモデルであれば用途は多い
  - 音声UIがChatGPTレベルの応答ができれば、お年寄りにも優しい機器に
  - 映像入力から何があるかを応答できれば視覚障害者に有用

JASA AI研究WGの紹介

AI研究WG

- 研究会とセミナーの２本立てで開催
- 研究会
  - 今年で4年目になるDeep Learningをすでに理解して開発できるメンバーが集まり、様々なテーマでAI活用研究を行う研究会
  - メンバーは現在8社17名
- セミナー
  - 今年で7年目になる初学者向けのDeep Learningセミナー
- AI研究WG発表会
  - 年度末に研究会/セミナー別で発表会を実施

研究会紹介

- エッジデバイス上でのDeep Learningの可能性や、様々なテーマで持続的に調査研究を行う
- 1ヶ月に1度、定例会議を開きDeep Learning周辺の最近の動向の共有、メンバーの研究内容の進捗発表を行なっている
- 全員でコンペに参加して実力を試したり
  - 個々のメンバーで興味のあるコンペに参加

セミナー紹介

- 1年間で3回の座学とグループでのDeep Learningデモ作成がゴール
- 講習にはGoogle Colaboratoryを利用
  - ColaboratoryはクラウドでJupyter Notebook環境なのでお手軽
- フレームワークはTensorflow＋Keras
- グループ間の情報共有、全体連絡にSlackを活用

研究会の個々の研究案件の紹介

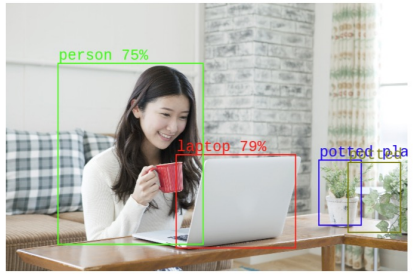
- 推論時の消費電力調査
- 競馬AI予測研究
- 低リソースデバイスAI
  - Edge TPUで推論
  - リザバーコンピューティング(ESN)の調査
- 異常音検出
- FPGA上での学習
- 強化学習
- JetBotで自動運転

推論時の消費電力調査

- 組込みAIの実用化に向け、Raspberry Pi4やEdge TPUで推論させた時の実消費電力を計測
  - Raspberry Pi4(人物検出)：4.2～6.0W
  - Edge TPU(物体検出)：5.2～6.0W

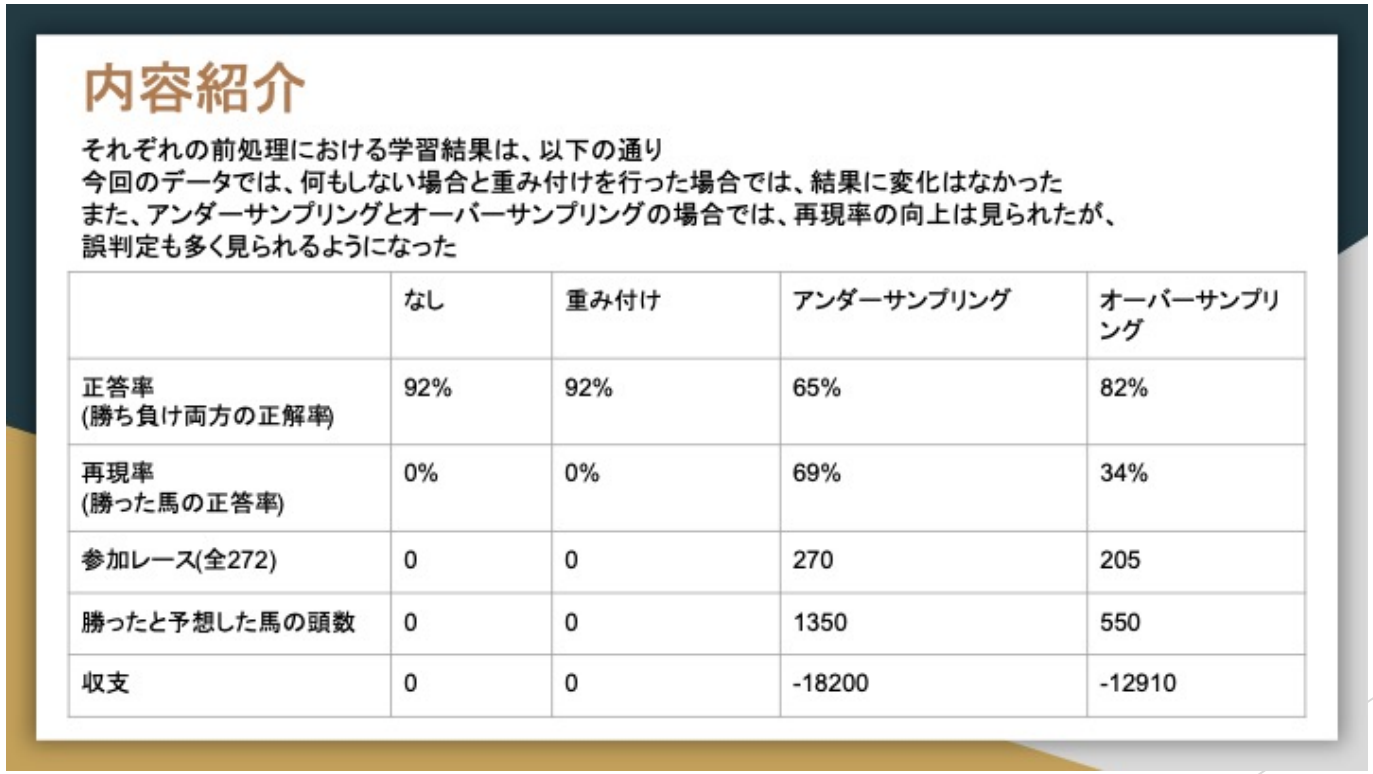
EdgeTPUでの物体検出

- 前掲の人物検出モデルがpycoralベースで動作しなかったため、EdgeTPUのexampleにあったMobileNetV2ベースの物体検出SSDをpycoralで動作
  - IDLE(EdgeTPU接続): 3.5W (接続のみで1.4W程度消費)
    - EdgeTPU単体: 0.3W
  - 推論: 5.2～6.0W 20fps
    - EdgeTPU単体: 1.0W～0.3W
  - 検出精度は低いが速度は速い
  - 思ったより消費電力は高くない



競馬AI予測研究

- 前年の研究から、前処理の導入による予測の改善を実施
  - 重みづけ・アンダーサンプリン・オーバーサンプリングを比較



内容紹介

それぞれの前処理における学習結果は、以下の通り
今回のデータでは、何もしない場合と重み付けを行った場合では、結果に変化はなかった
また、アンダーサンプリングとオーバーサンプリングの場合では、再現率の向上は見られたが、
誤判定も多く見られるようになった

	なし	重み付け	アンダーサンプリング	オーバーサンプリング
正答率 (勝ち負け両方の正解率)	92%	92%	65%	82%
再現率 (勝った馬の正答率)	0%	0%	69%	34%
参加レース(全272)	0	0	270	205
勝ったと予想した馬の頭数	0	0	1350	550
収支	0	0	-18200	-12910

低リソースデバイスAI

- インテルのEdge TPUを試してみる
  - 性能が高くもっと掘り下げてみる価値ありと判断

Edge TPU 推論性能比較

model: Semantic Segmentation

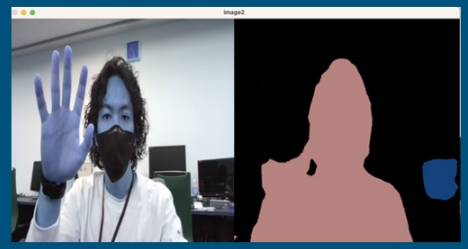
Mac CPU
- FPS: 1

Mac Radeon Pro 555X 4 GB
- FPS: 3.70

EdgeTPU:
- FPS: 17.48

GPU

TPU

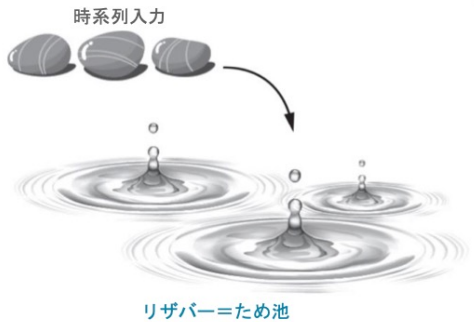


低リソースデバイスAI

- リザバーコンピューティング(ESN)の調査
  - リカレントニューラルネットワークの特殊なモデルを一般化した概念で、時系列情報処理に適している

概要

リザバーコンピューティング

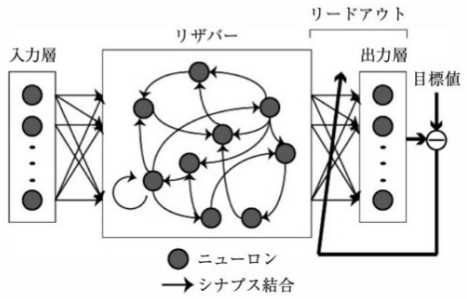


リザバー＝ため池

石（入力） を入れると波紋が生じる
波紋は、石の大きさや形、投げ入れる順番によって異なる
波紋を見ればそこから入力情報について分析可能

代表例

Echo State Network



2001年にJaeger（イェーガー）によって提案
リカレントニューラルネットワークの特殊なモデル
出力層の重みのみを学習の対象（他は固定）
→低リソース環境で活用可能性あり

https://www.skillupai.com/blog/tech/reservoir-computing/

FPGA上での学習

- FPGAでの学習の可能性、限界、課題の調査を行う
  - 去年度はまず推論をSignateのエッジAIコンテストに参加して試してみた

AIエッジコンテスト取組内容

DeepLabV3の量子化モデルをFPGAで動作

- 前述の内容を考慮し、推論用のpythonスクリプトを作成して動作させた
- FPGAボード：KV260
- 入力：192x120x3
- 実行速度：5fps

入力画像

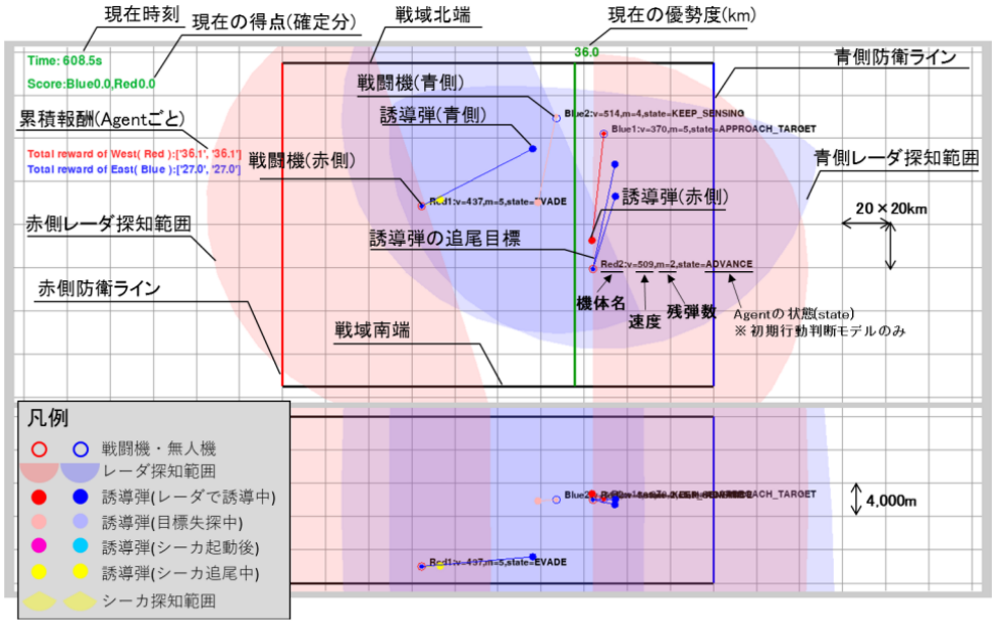
可視化画像

PointPainting全体を動かすところまではいけなかった

強化学習

- 強化学習を主題としたシミュレーションコンペへの参加
  - 空戦AIチャレンジに再挑戦

コンペの概要③

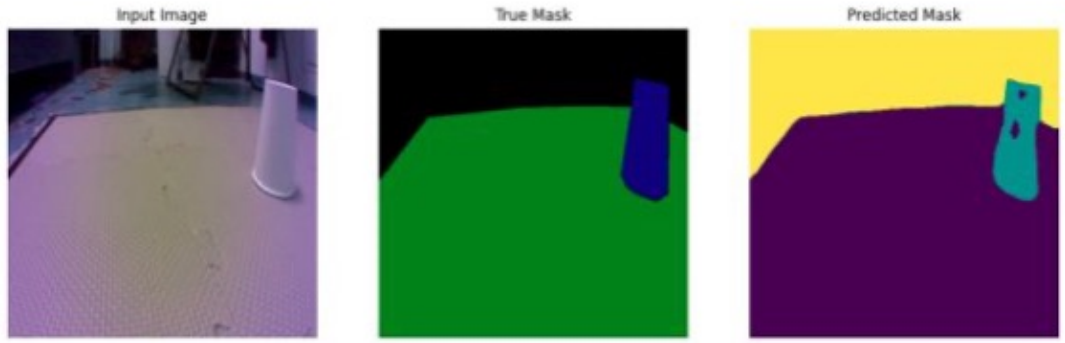


JetBotで自動運転

- ラインをトレースして、ライン上の障害物を検知し停止するのを目標とした
  - セマンティック・セグメンテーション モデルの学習と、ライントレースのお試しまで完了

セマンティックセグメンテーション

学習結果



入力画像(左)、正解画像(中央)、予測画像(右)