FUJITSU
shaping tomorrow with you

スマホながら歩き検出システム

～エッジAIからクラウドまでIoTシステムをモデル駆動開発～

2019年7月18日
富士通コンピュータテクノロジーズ
オートモーティブテクノロジ事業部先行基盤開発部
石田 晴幸

テーマと目的

FUJITSU

- 目的
  - エッジAIを含むIoTシステムを開発し技術を獲得する
- テーマ
  - エッジAI・クラウド連携なIoTシステムで、スマホながら歩きを検出・警告する
- 獲得する技術
  - ラズパイ+ Intel Neural Compute Stick2 (NCS2)でエッジAI
    - NCS2でOpenPoseによる人体骨格推論を行う技術
    - 骨格推論結果から「スマホながら歩き」を検出するAI開発技術
  - Azure App Serviceのよるサーバレスwebアプリ開発
    - 骨格推論結果の収集
    - 「スマホながら歩き」検出ＡＩの再学習・配付を行える枠組み
  - IoTシステムの論理設計・物理配置をモデル駆動開発する技術
    - システムの構成要素の物理配置はモデルからコード生成

なぜラズパイ+NCS2　(エッジAI)

FUJITSU

■ 弊社・富士通のAI実行環境への取組み

項番	ジャンル	取組み
1	クラウド	Zinrai ディープラーニング
2	FPGA	2値化/3値化により推論論理を高度圧縮
3	推論用デバイス	私がやろう！ SWESTで展示もしよう！

■ 推論用デバイス

項番	推論用デバイス	商品名
1	NVIDIA GPU (多数)	NVIDIA Jetson TX2 他
2	Intel Movidius Myriad X VPU	Intel Neural Compute Stick 2 (NCS2)
3	Google Edge TPU coprosessor	Google Coral Edge TPU USB Accelerator

2019年4月の時点で価格・入手性・ソフトウェア開発の容易さからNCS2を選択。
NCS2では学習済みの人体骨格推論モデルが使える。

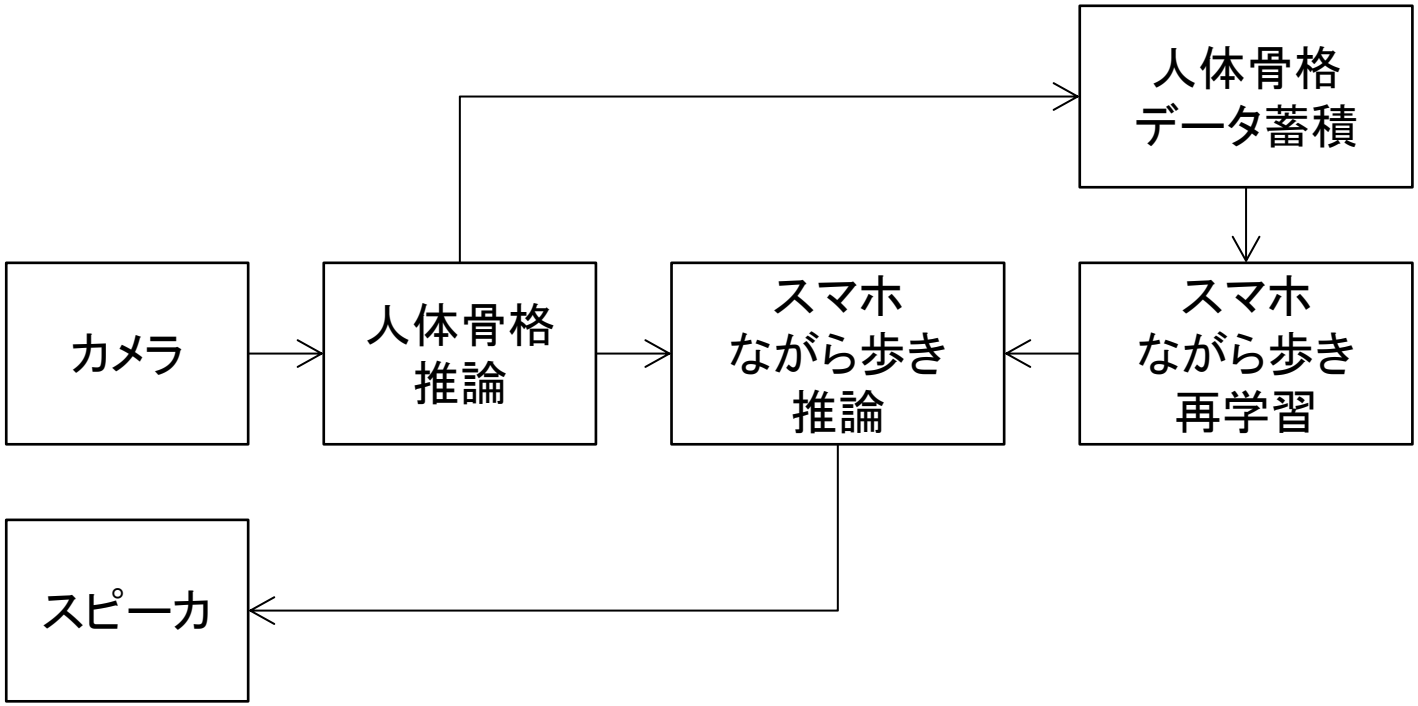
なぜAzure App Service (クラウド)

FUJITSU

項番	クラウドサービス	コメント
1	FUJITSU Cloud Service for OSS Zinrai ディープラーニング	Linuxの構築から行う必要がある Zinraiディープラーニング上でGPUを使い TensorFlow機械学習ができる
2	Azure App Service Azure Nシリーズ	Python/Flaskが動く環境が提供される Nシリーズ上でGPUを使いTensorFlow機械学習ができる。
3	AWS Lambda, Amazon EC2 P2	AWS Lambdaの上にFlaskをインストールして使用できる。 P2上でGPUを使いTensorFlow機械学習ができる。

- Pythonで処理を書けば、ラズパイでもクラウドでも動作させられる。
  ⇒どの処理をどこで行うか、物理配置を後から変更しやすい
  上記クラウドはいずれもPythonで処理を記述可能。
- NCS2/Google Edge TPU/Jetsonを見据えるとTensorFlowを使いたい。
  上記クラウドはいずれもGPUを使った機械学習が可能。
- Webアプリの取り掛かり易さからAzure App Serviceを選択

なぜモデル駆動開発



システムの論理的な構造

- PoCシステムを組むにも、上図の要素を物理的にどう配置すると良いかはやってみないとわからない
  （しかも最適解は時代の変化ですぐ変わる）
- ならば論理設計と物理配置を分離して、物理配置はツールにやらせよう
  (Autosarの考え方のパクリ)

なぜスマホながら歩き検出

FUJITSU

- 私が、駅のホームや階段でスマホながら歩きする人に不満
  - 自分が必要と思うものを作ることが、モチベーション維持に必要
  - 物体検出はもはやありふれた技術。今後は人の行動解析が重要

イラストは https://www.irasutoya.com/ より引用

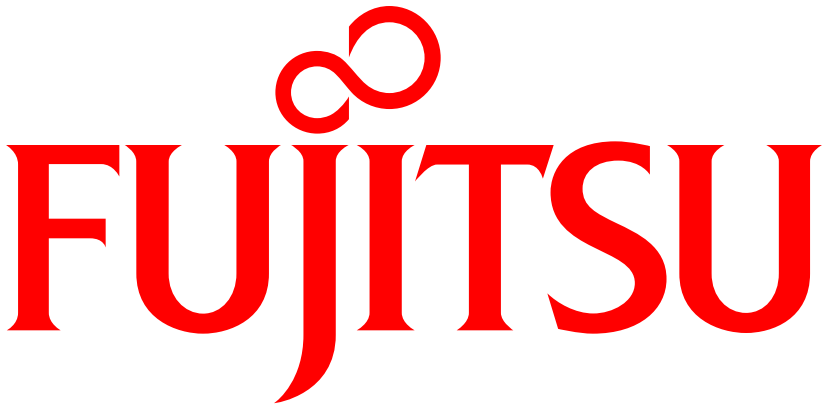
デモ

FUJITSU

- 会場では、実際に動作するデモをお見せする予定です。

hpeex02_2.py ▸ ...

```
30  LINE_WIDTH_BONE = 2
31  THRESHOLD_KEYPOINTS = 0.1
32
33
34  class ModelDesc(object):
35      def __init__(self, device_name, model_xml, weight_bin, num_requests=1):
36          self.device_name = device_name
37          self.model_xml = model_xml
38          self.weight_bin = weight_bin
39          self.num_requests = num_requests
40
41
42  class HumanPoseEstimator(object):
43      def __init__(self, model_desc: ModelDesc):
44          plugin = IEPlugin(device=model_desc.device_name)
45          self.net = IENetwork(model=model_desc.model_xml, weights=model_desc.weight_bin)
46          self.exec_net = plugin.load(network=self.net, num_requests=model_desc.num_requests)
47          blob_input = self.net.inputs[BLOB_NAME_INPUT]
48          self.size_model_in = (blob_input.shape[3], blob_input.shape[2])
49          blob_pcm = self.net.outputs[BLOB_NAME_PCM]
50          blob_paf = self.net.outputs[BLOB_NAME_PAF]
51
52
53      def __convert_image_to_inputblob(self, image):
54          w = image.shape[1]
55          h = image.shape[0]
56          w_model_in = self.size_model_in[0]
57          h_model_in = self.size_model_in[1]
58          scale_w = w_model_in / w
59          scale_h = h_model_in / h
60          scale = min(scale_w, scale_h)
61          w_scaled = int(w * scale)
62          h_scaled = int(h * scale)
63          print('w:{0}, h:{1}, w_model_in:{2}, h_model_in:{3}, w_scaled:{4}, h_scaled:{5}'.
64                format(w, h, w_model_in, h_model_in, w_scaled, h_scaled))
65          image_scaled = cv2.resize(image, (w_scaled, h_scaled), interpolation=cv2.INTER_CUBIC)
66          w_model_in = self.size_model_in[0]
67          h_model_in = self.size_model_in[1]
68          image_in_size = np.full((h_model_in, w_model_in, 3), 128)
```



shaping tomorrow with you