技術者が知っておきたいDeep Learningの基礎と組込みでの利用
～今さら聞いてくださいDeep Learning～

SWEST18 2016/8/26

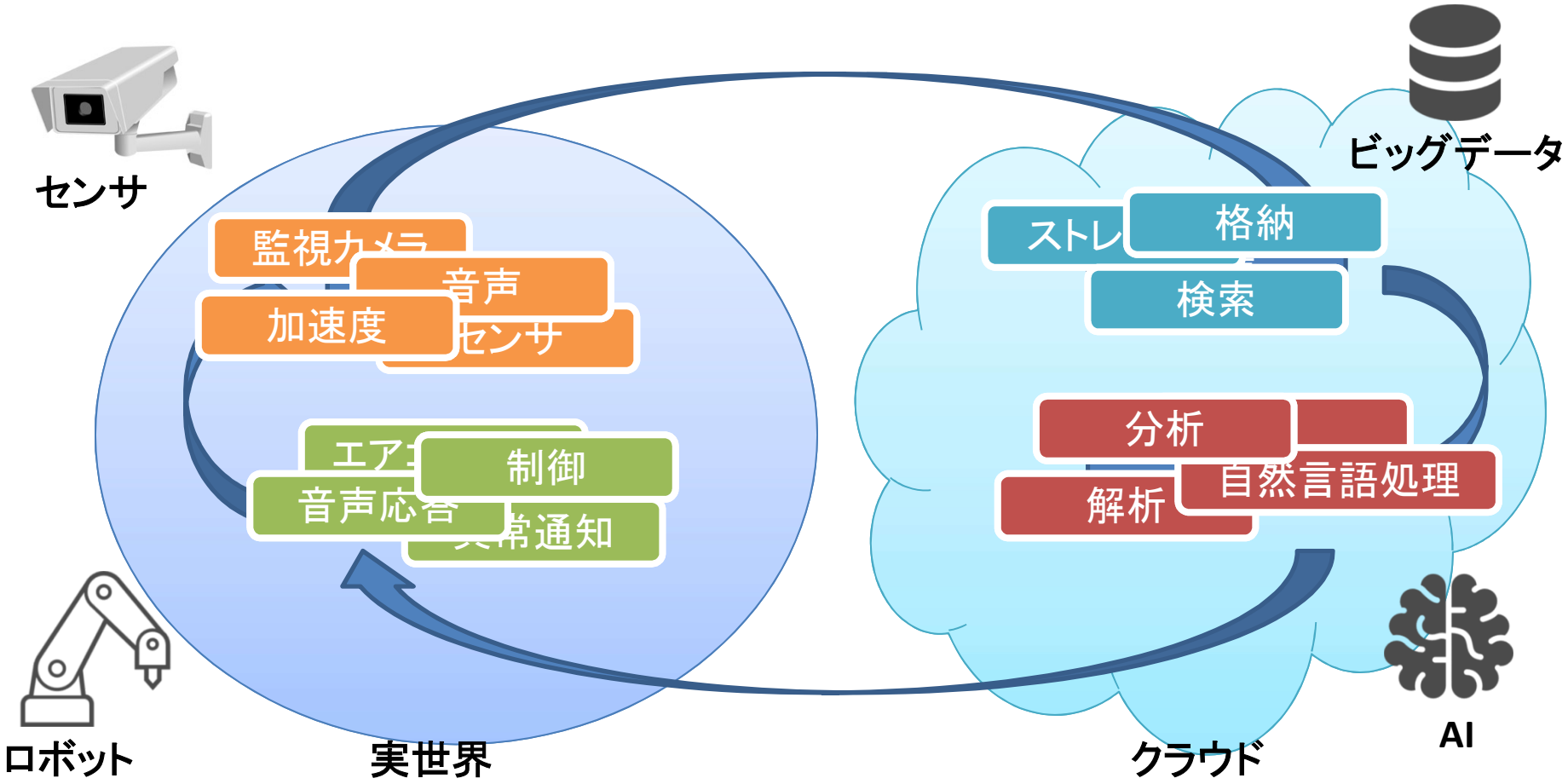
(株)システム計画研究所
上島 仁

Agenda

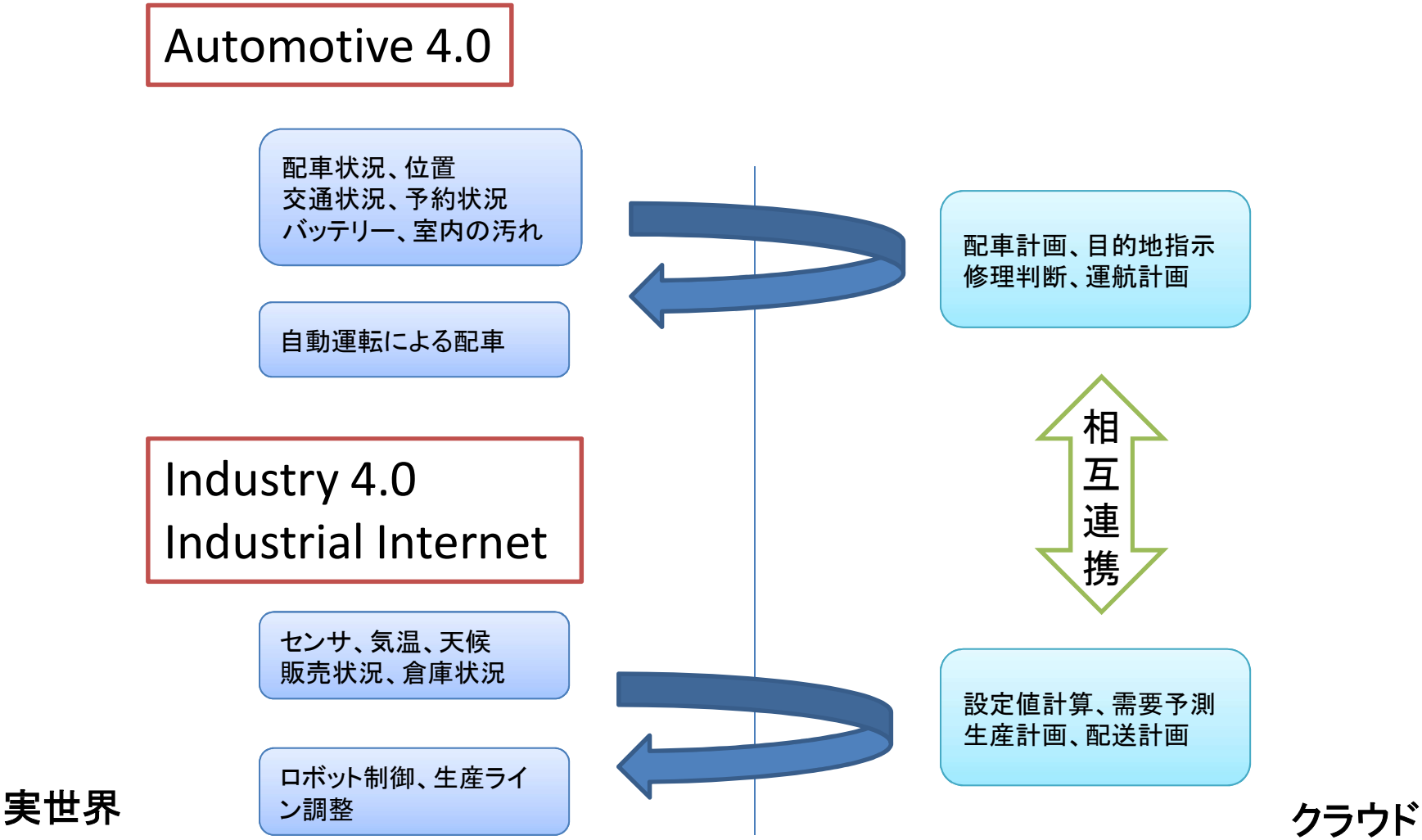
1. IoTを取り巻く世界観
2. Deep Learning技術解説
3. 事例紹介
4. 開発の実際
5. 組込みでの利用

1. IoTを取り巻く世界観

データ駆動型社会


センサ
ビッグデータ
監視カメラ
音声
加速度
センサ
エアコン
制御
音声応答
異常通知
ストレージ
格納
検索
分析
自然言語処理
解析
ロボット
AI
実世界
クラウド

サービスの例

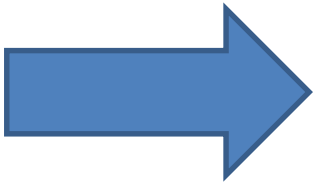

Automotive 4.0
配車状況、位置
交通状況、予約状況
バッテリー、室内の汚れ
自動運転による配車
配車計画、目的地指示
修理判断、運航計画
相互連携
Industry 4.0
Industrial Internet
センサ、気温、天候
販売状況、倉庫状況
ロボット制御、生産ライン調整
設定値計算、需要予測
生産計画、配送計画
実世界
クラウド

キーワード

- IoT
  - 実世界におけるデータ収集
- Big Data
  - データの蓄積
- ロボット
  - 実世界へのアウトプット
  - 物理的サービス
- AI
  - データの分析、意味を取り出す
  - クラウド上のみならず、全体に遍く存在

データ駆動型社会におけるAIの位置づけ

- クラウド上
  - IoTで集まるデータを価値に変換する
    - 大量に蓄積されたデータの中から、重要な情報を抽出する
- エッジ
  - ネットワーク上のデータを削減する
    - データ量の爆発的増加に比べ、通信帯域はそれほど増えない
    - 分析に必要なデータは残しつつ、ノイズとなるデータを削減
  - データの補完
    - 故障したセンサデータの推定
    - センサの数を抑える



意味≒価値の取り出し

コスト削減

人工知能とは

人工知能 (artificial intelligence)

人工的にコンピュータ上などで人間と同様の知能を実現させようという試み、或いはそのための一連の基礎技術を指す。

人工知能研究の目的

- 知性を持つ機械をつくる
- 人間が知能を使って行う作業を、機械で代替する

強いAIと弱いAI

区分	キーワード	応用事例	特徴
強い(Strong)AI	人間的 前頭葉 知性	鉄腕アトム？	そもそも知性とはなに？ 思いやりや共感？
弱い(Weak)AI	動物的 小脳 情報処理	画像認識 異常検知 自動運転 Alpha GO りんな	・コグニティブコンピューティング ・人の手に余る大量のデータを 瞬時に処理 ・非構造のデータを分析 画像、音声、自然言語、 センサデータ等 ・特定の問題セットに特化

2. Deep Learning技術解説

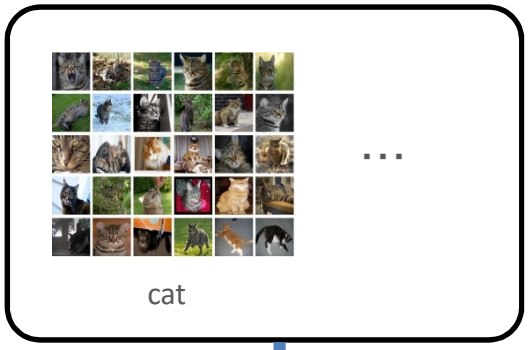
機械学習のイメージ

- 機械学習とは「明示的にプログラミングすることなく，コンピュータに行動させるようにする科学」（Andrew Ng）のこと。一般には過去データやサンプルから学び認識や予測、分類を実現する技術。

教師付き機械学習の例

$\{(x_i, y_i), i = 1,...N\}$

大量のラベル付き学習データ
（x:画像、y:ラベル）



入力

出力

“cat”

未知のデータ

学習データに含まれていない未知のデータを、正しく認識させることが目標

機械学習の分類

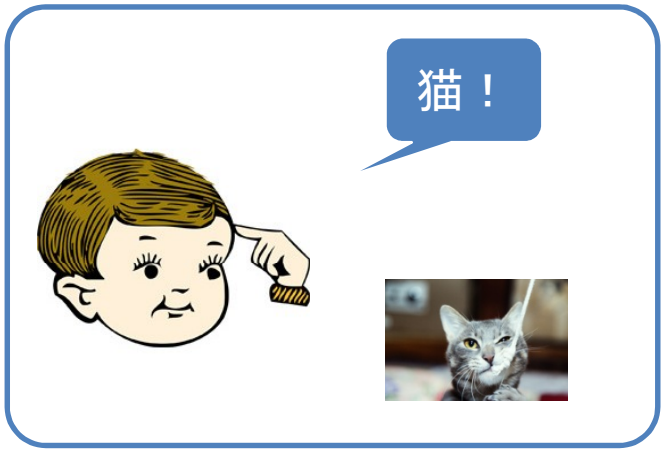
- 教師あり学習
  - 既知の情報からパターンを学習
  - 予測対象が数値であれば「回帰」、カテゴリであれば「分類」
- 教師なし学習
  - データのパターンをデータのみから推測
  - クラスタリング
- 他にも、半教師あり学習、強化学習、等々

色々な機械学習

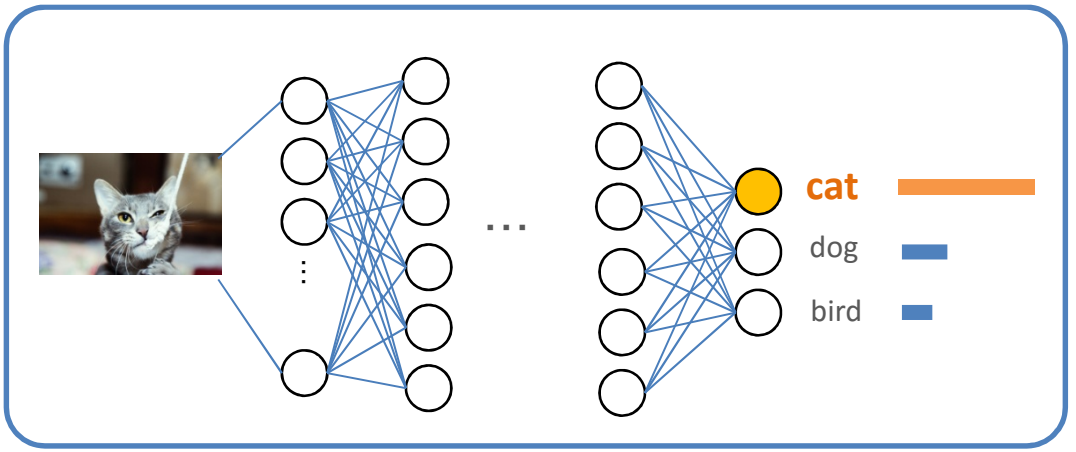
手法	概要
ロジスティック回帰	1950年代から存在する確率モデルで2値分類問題を解く手法。統計的手法とも言えます。
決定木・回帰木	分類や回帰問題を解く予測モデルを決定木の形で学習する。 条件分岐で予測値を求めるため、人の目で見てモデルが分かりやすい。
ブースティング	一連の弱い仮説器をまとめることで強い仮説を生成する集団学習アルゴリズム。教師あり学習に対応する
Random Forest	決定木を弱仮説器とする集団学習アルゴリズムです。
SVM（support vector machine）	1990年代一世を風靡した手法。Deep Learningが登場する以前は最も認識性能が優れ手法の1つだった。現在もベンチマーク的な意味合いを持っている。
Deep Learning	1960年代から存在するニューラルネットワークを用いた手法。注目・幻滅を繰り返しきましたが、近年計算機パワーの増大と過学習を抑える工夫があいまって画像認識、音声認識、動画認識など様々な分野で高い精度を叩き出しています。現在 最も認識性能が優れた手法と言われています。

Deep Learningのイメージ

多層ニューラルネットを用いた人工知能構築技術の総称



人間の脳による判別

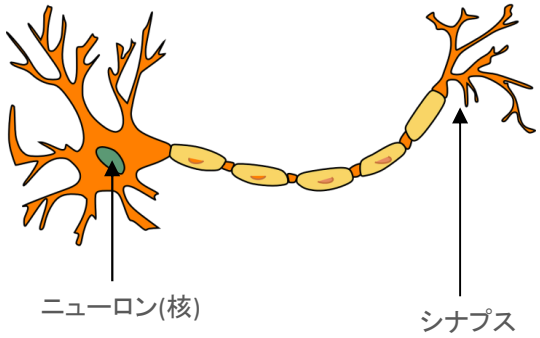


Deep Learningによる判別

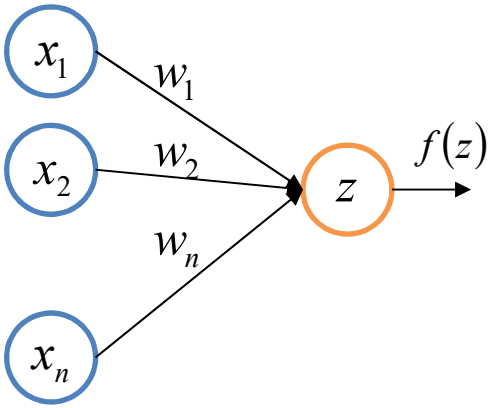
ニューラルネットワーク

- 脳神経を計算機上でシミュレーションすることを目指した数学モデル

神経細胞(ニューロン)



単純パーセプトロン(1960年代～)



入力値の線形結合

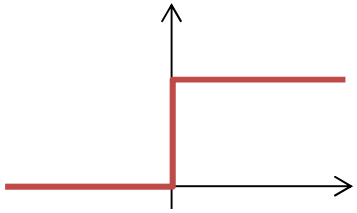
$$z = \mathbf{w}^t \mathbf{x} - T$$

結合重み　バイアス

活性化関数

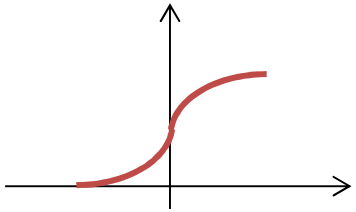
ステップ関数

$$f(z) = \begin{cases} 1 & z > 0 \\ 0 & otherwise \end{cases}$$



シグモイド関数

$$f(z) = \frac{1}{1+\exp(-z)}$$

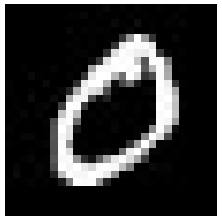


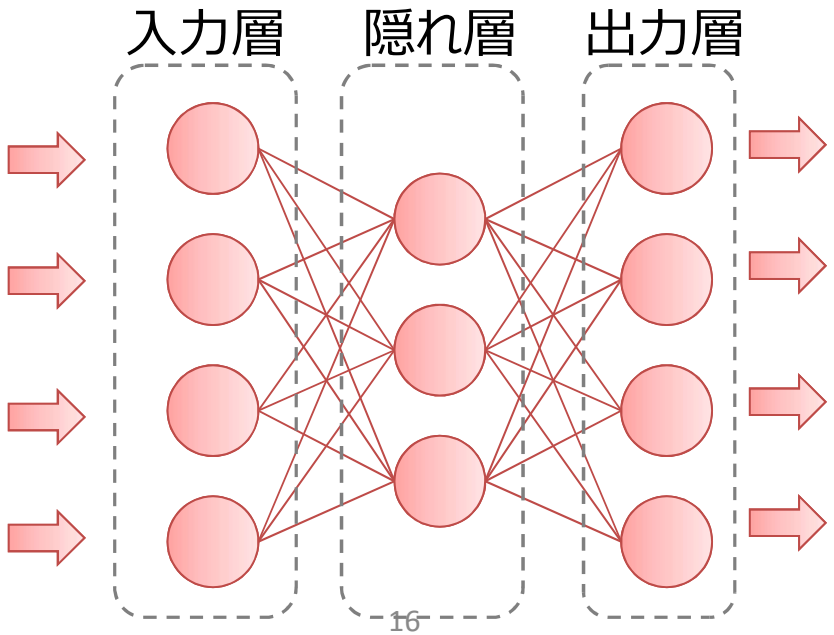
ニューラルネットワークの動作

- 多数の人工ニューロンから構成された層を何層も重ね合わせた多層で構成
- 人工ニューロンは与えられた数値データを発火関数にかけ、発火の閾値を超える場合にはニューロンとニューロンを繋ぐ重み付きのシナプスを通して次の層のニューロンに情報を伝達します
- 学習によりシナプスの重みの値が更新され、特定の特徴を持ったデータを入力層に与えられた時に反応するネットワークが構築されます

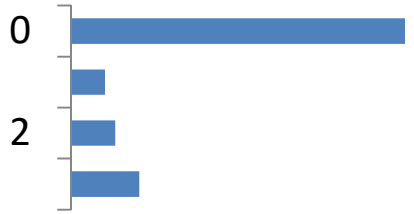
ニューラルネットワークの図例（画像認識）

入力層の各ニューロンに画像のピクセルごとのデータを入力します



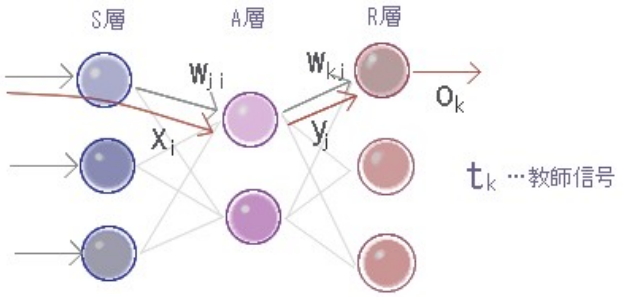


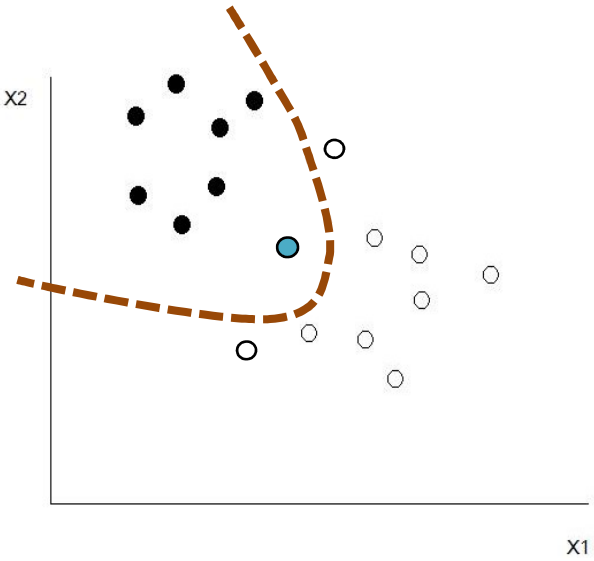
入力した画像の分類に対応した出力層のニューロンが最も高い数値を出力します



機械学習の発展～Deep Learningの登場

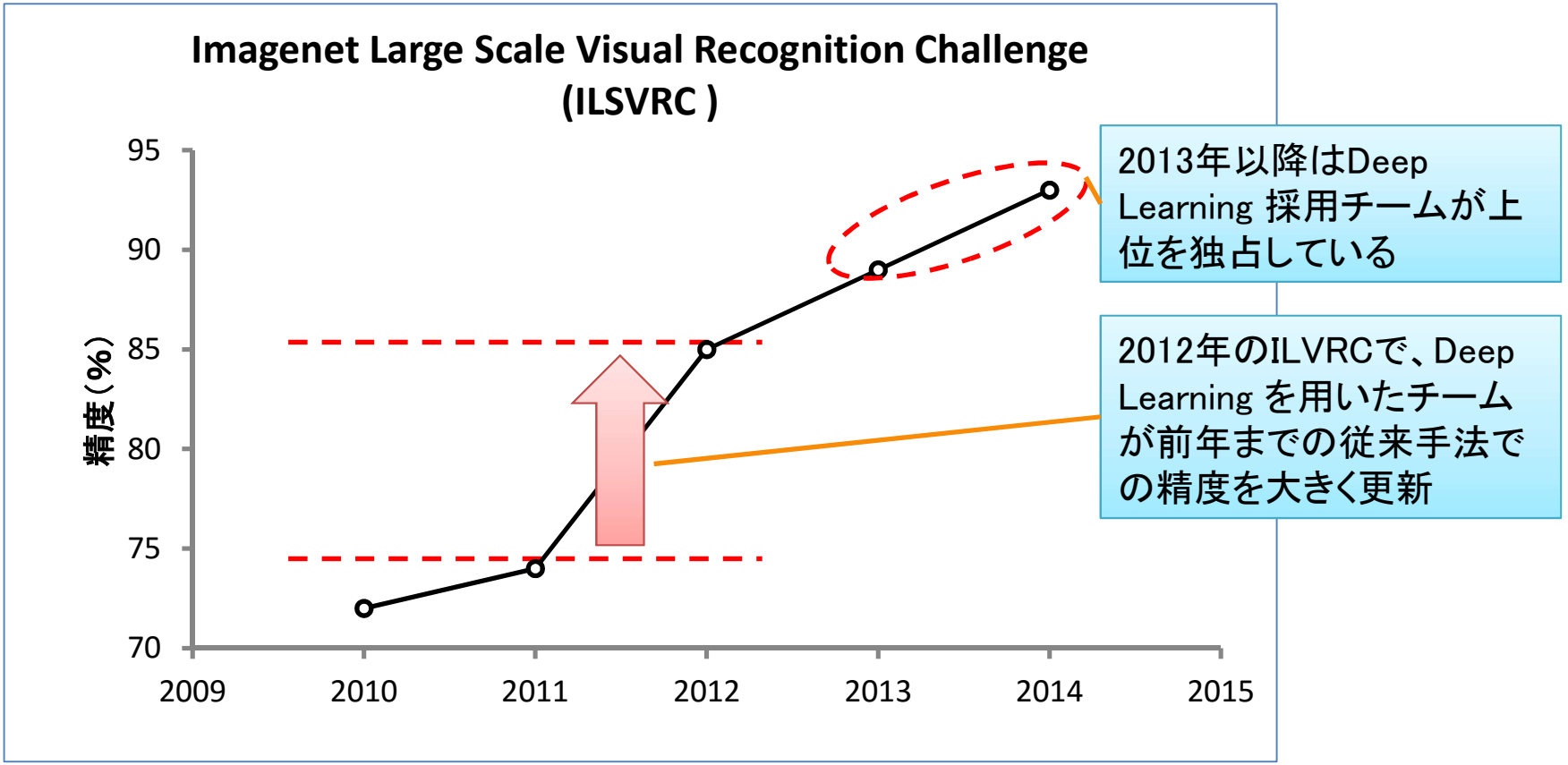
- 1960頃～
  - パーセプトロン（2層）
    - 注目を集めるもXORが解けないことわかり下火に
- 1980頃～
  - 多層パーセプトロン（3層）
    - **バックプロパゲーション**の発明
    - **課題** →学習が収束しない、過学習となる
- 1990頃～
  - SVM, Boosting, Random Forest, ...
- 2006頃～
  - Deep Learning
    - Layer-wise Training　2006年（Hinton先生の有名な論文）
      - Deep Neural Networkの学習が**過学習してしまう問題**を解決
      - 従来よりも**多層のニューラルネットワーク**を扱えるようになった
      - 画像認識、音声認識、自然言語処理などさまざまな分野で圧倒的な性能を達成





Deep Learningの開花

- 各種コンテストで高い精度を叩き出し、現在では注目を集める技術となった
- Deep Learning は大量の学習データから**特徴自体を学ぶことができ**、AIにおける30年来のブレークスルーを果たしたとも言われている。



なぜ、今Deep Learningなのか？

ニューラルネットワークの根本原理が変わった訳ではない

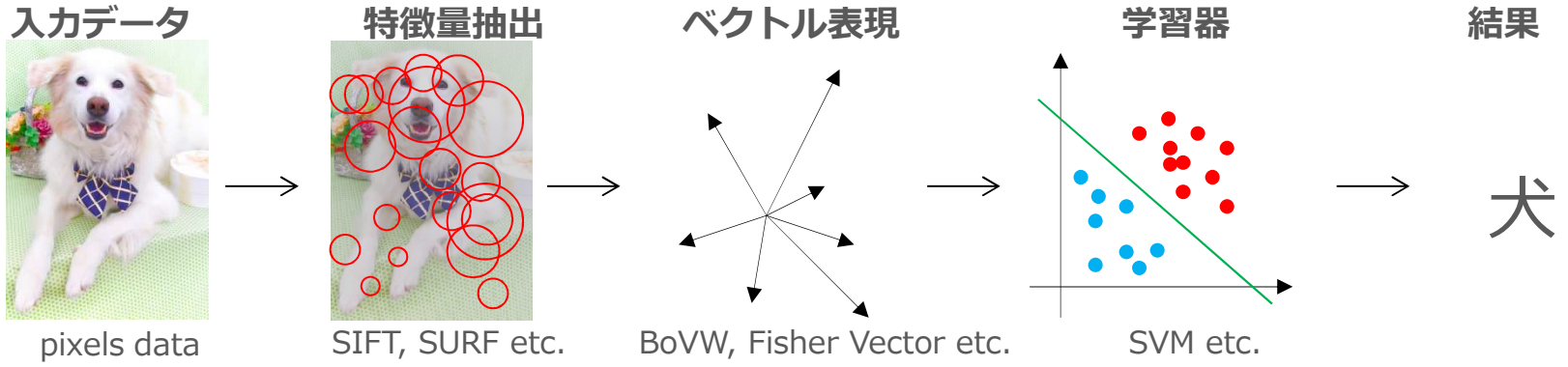
■問題点

- 学習が収束しない
  - 入力に近い層の学習が遅い
  - 層を遡る過程で誤差が多数のニューロンに拡散されてしまい、パラメータがほとんど更新されない
- 過学習しやすい
  - 訓練データのみに過度に適応

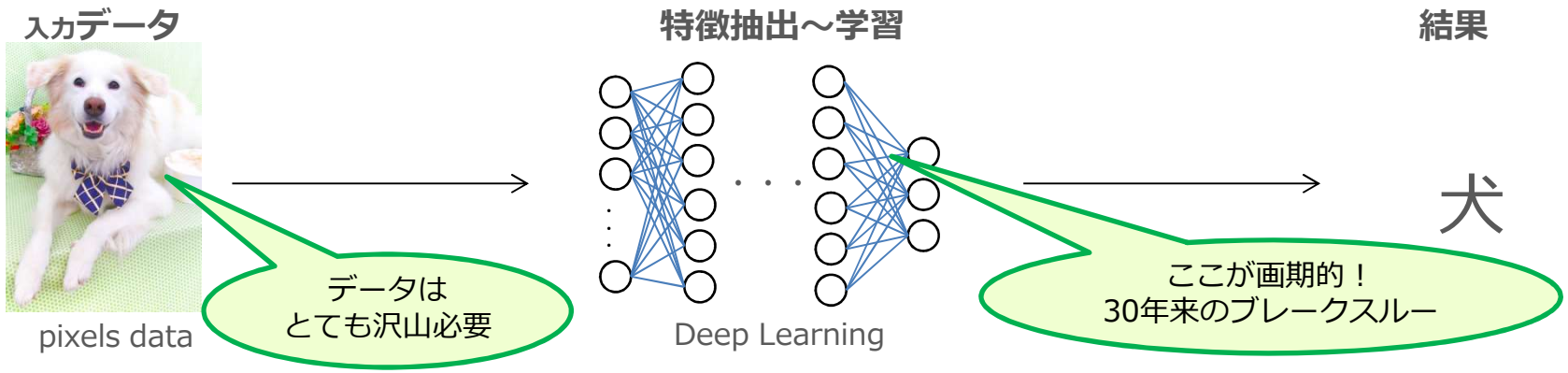
1. 活性化関数 ReLUの発明
2. 過学習回避手法の発明
3. その他、最適化に関する手法の発達
4. データの増加
5. 計算機の高速化(GPGPUの恩恵)

従来手法との比較

機械学習（従来手法）



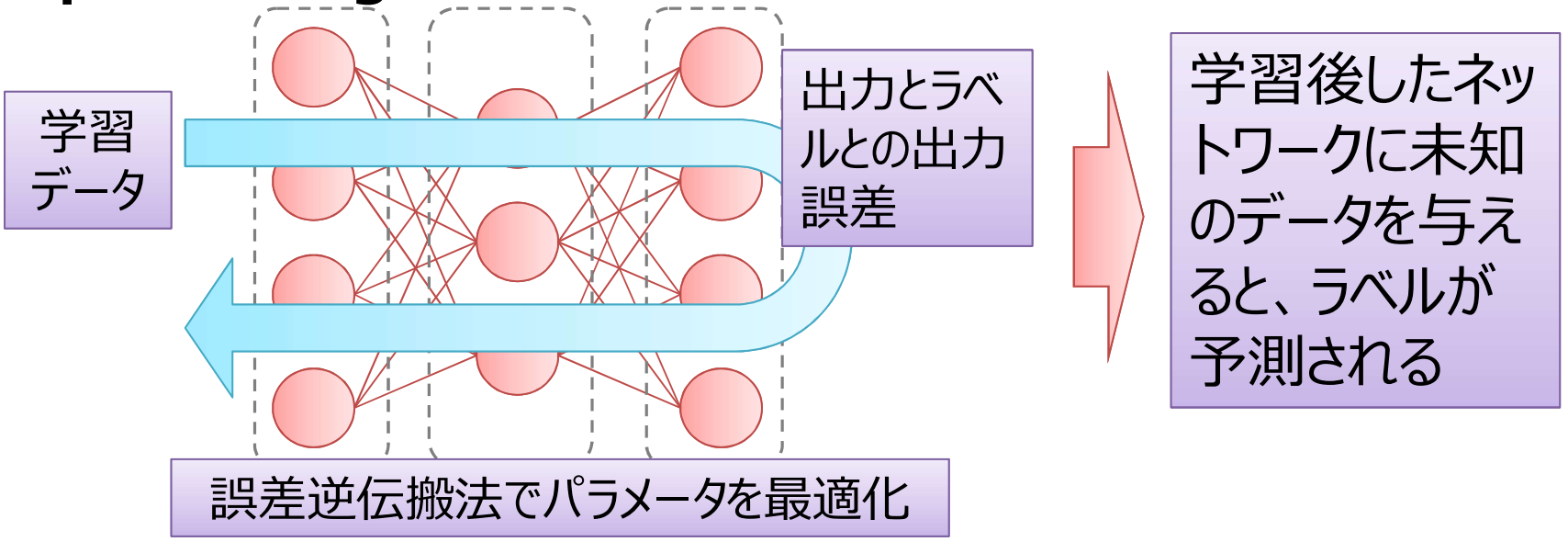
Deep Learning（深層学習）



Deep Learningの学習

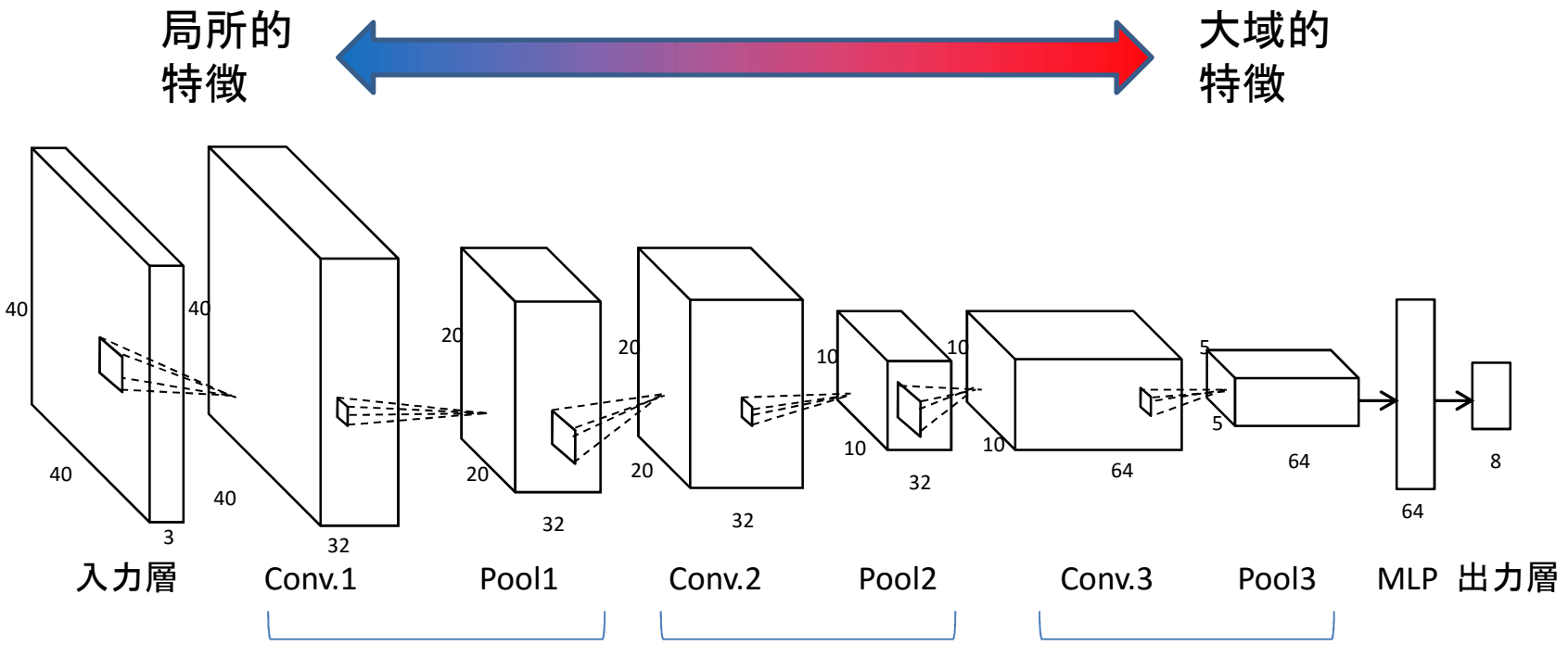
- Deep Learningの学習では、まずネットワークに学習データを入力し、その後出力値とラベルとの誤差が少なくなるように誤差逆伝搬法（バックプロパゲーション）でネットワークの重みパラメータを**最適化**します
- 従来のニューラルネットワークと比較して、Convolution、Poolingなどの技術が使われることで、局所解に陥らない工夫がされています

Deep Learningの学習のイメージ：

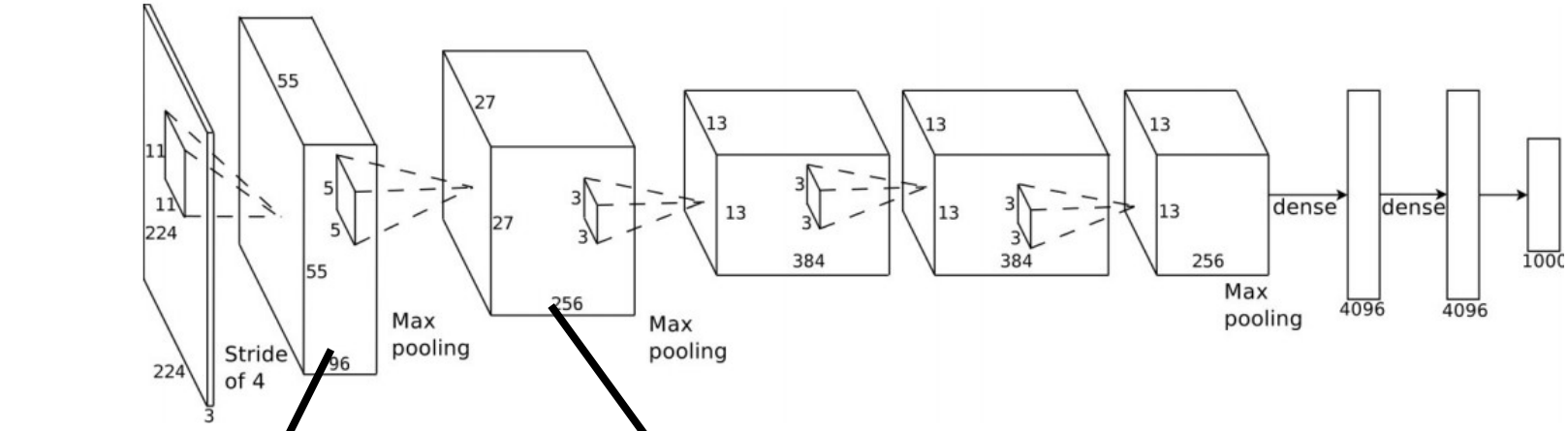


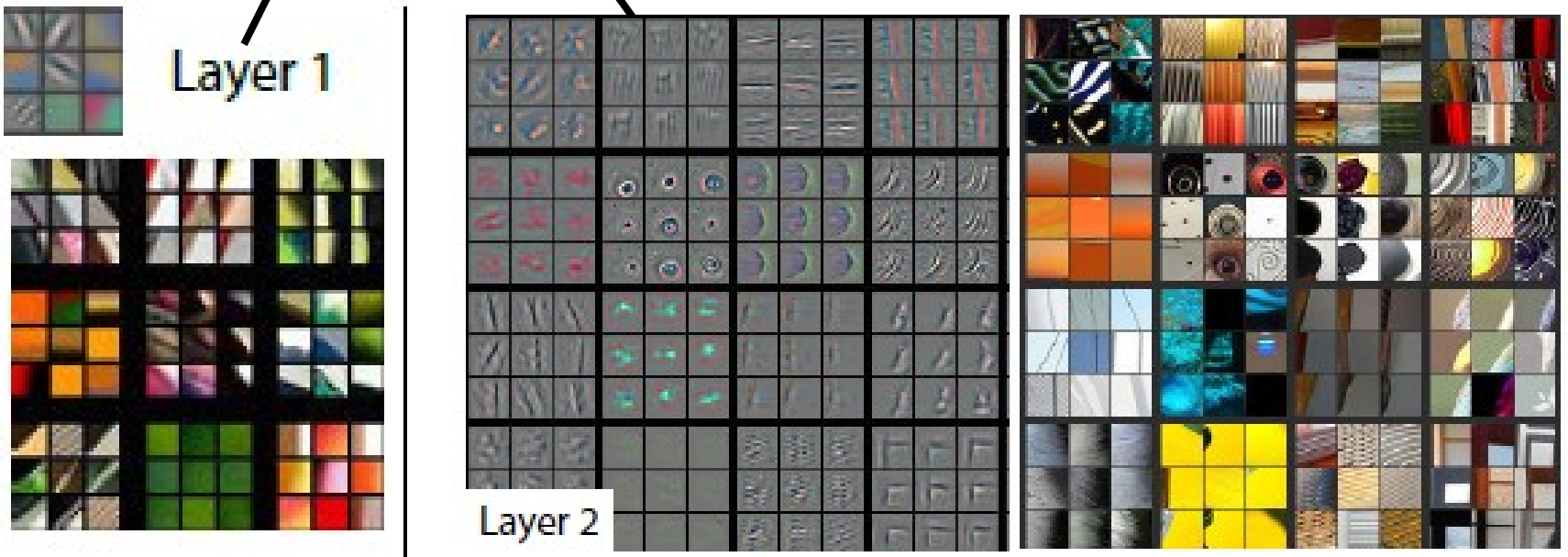
畳み込みニューラルネット(CNN)

- 畳み込み層とプーリング層と呼ばれる2種類の層を交互に積み重ねた構造を持つ
- ネオコグニトロン(1979, 福島邦彦)
  - 第一次視覚野の、S 細胞, C 細胞をモデル化
- 画像認識で高い性能を示す
- 画像フィルタを連続的にかけるイメージ

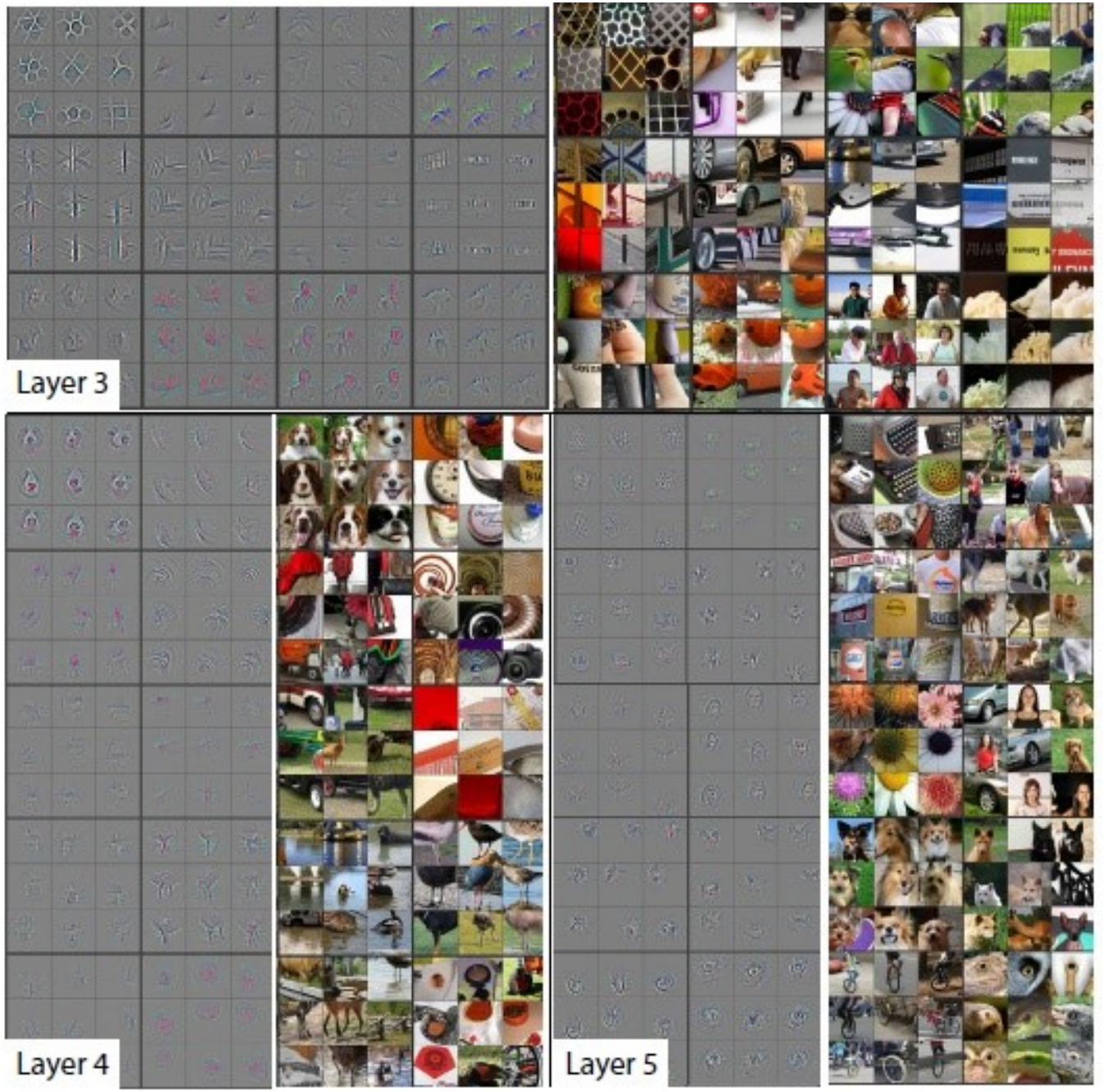


各層で反応する形状


11
11
224
224
3
55
5
5
55
96
Stride
of 4
27
27
3
3
256
Max
pooling
13
13
3
3
384
Max
pooling
13
13
3
3
384
13
13
256
Max
pooling
dense
4096
dense
4096
1000

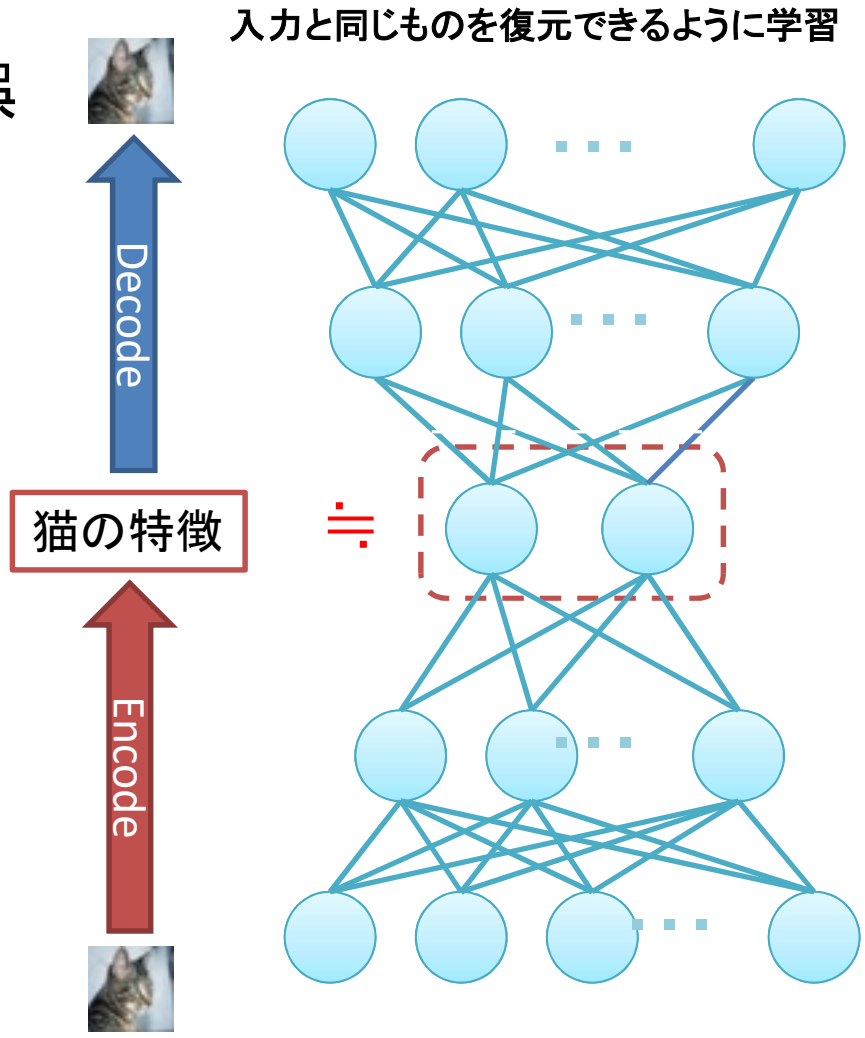

Layer 1
Layer 2

各層



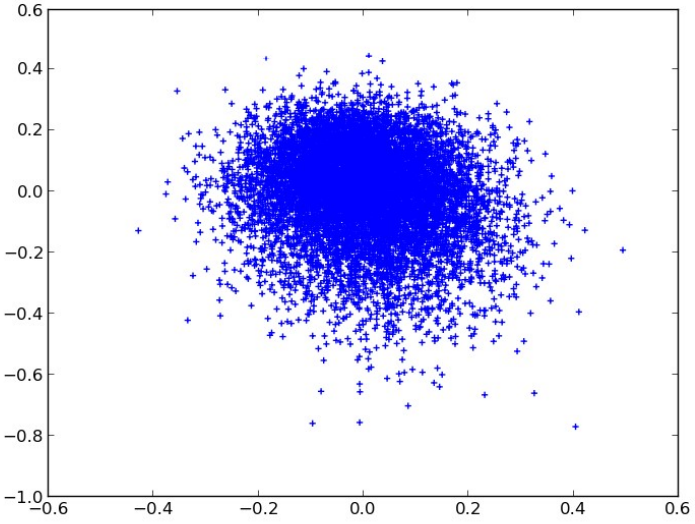
Stacked AutoEncoder

- 次元圧縮後に再度元に戻し、誤差が少なくなるように学習
  - 中間層に特徴が凝縮される
- 情報圧縮が可能
  （上手くやれば）

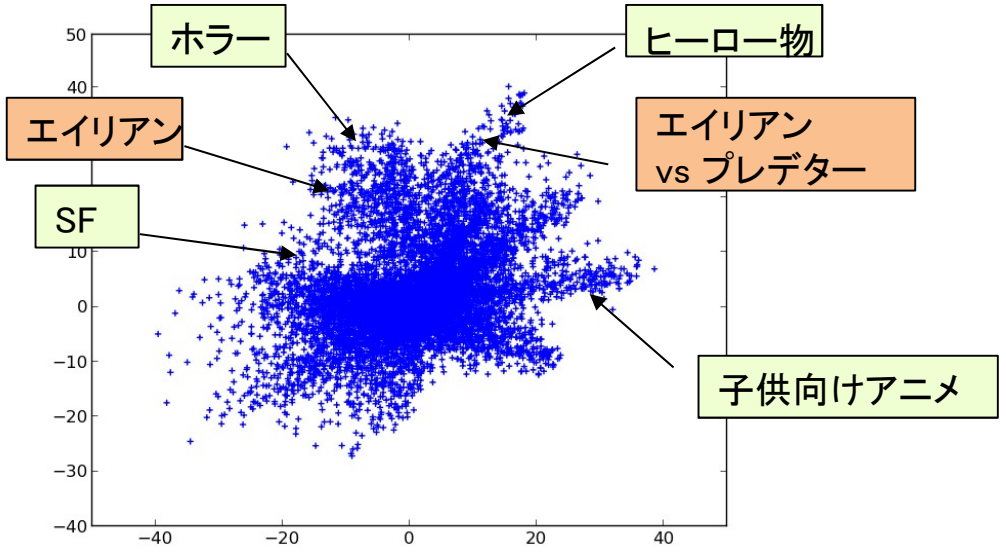


Stacked AutoEncoder --- 従来手法との比較

- Movie Lensプロジェクト TagGenome
  - 映画の特性や感想、原作者等でタグ付け
  - タグ例：animal、fun、G.Orwell等、1000種類



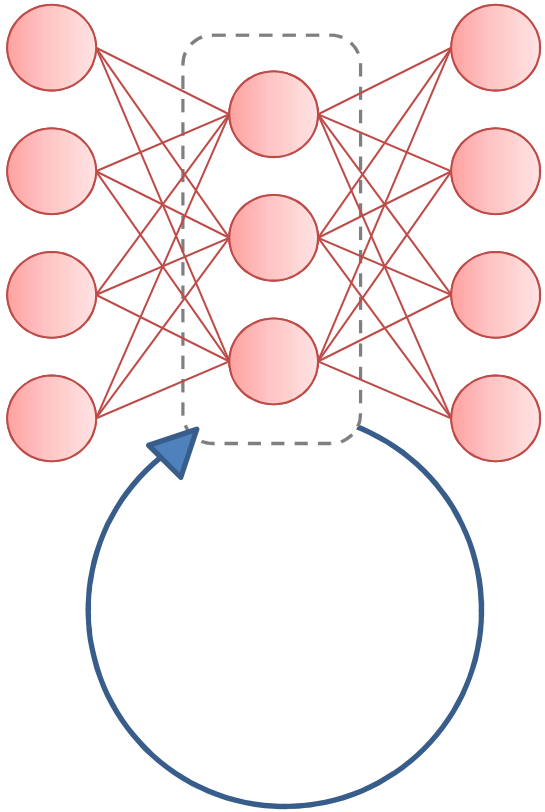
PCA



Autoencoder

RNN (Recurrent Neural Network)

- 再帰型ニューラルネットワーク
- 不定長系列データを扱うための手法
- 時刻t-1とtのデータの相関を学習可能
  - Xの次にはYが来る可能性が高い、というような場合
- 応用例
  - 自然言語処理
  - 音声認識



私	は	Deep Learning	を	勉強して	います
1	2	3	4	5	6

Deep Learningの特徴 (まとめ)

- 特徴量を設計する必要がない
  - 適切な特徴量を獲得できるため、精度が高い
- 他の機械学習アルゴリズムに比べ、多くの学習データが必要

- BlackBox性が高い
  - 学習の過程で何が行われているか、見えない
  - ただし、Uncontrollableというわけではない
    - 特徴量の設計はできないが、学習データの設計は可能
  - 学習済みのネットワークから、モデル(数式)の取り出しができない

- 学習には泥臭い作業が必要

3. 事例紹介

郵便番号認識

- LeNet (1998)
  - Deep Learningが実用化された最初のケース
  - LeCun による手書き郵便番号認識への応用
  - Convolutional Neural Network (畳み込みニューラルネットワーク) を利用

自然言語処理

- SmartNews
  - 連日1000万件以上の記事のトピックを自動分類
  - 記事タイトルと本文からDeep Learningで特徴量を抽出、10数個のカテゴリに自動分類
- ニュース画像の判別(CNN)と組み合わせた例もあり
- 検索エンジンの中身でも、Deep Learning関連技術が徐々につかわれ始めている

音声認識

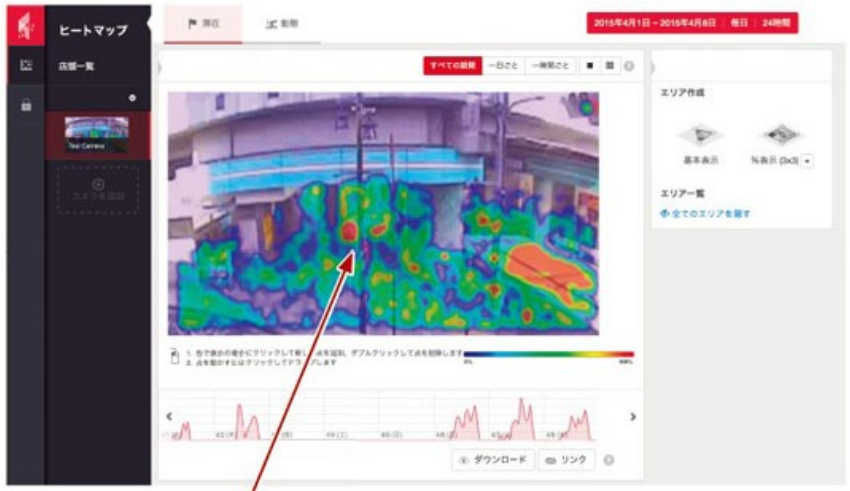
- YJVOICE (Yahoo! Japan)
  - 2015年5月からサービス提供開始
  - 音声検索クエリから4年半ほどで蓄積した音声データを利用
    - 専門スタッフが検聴して、発音ラベルを付与
    - カーナビでの使用を想定し、自動車内での雑音データを計算機上で加算
  - 1000時間強の音声データを用いて学習に1か月ほどかかった

金融分野 (FinTech)

- Capitalico (ALPACA)
  - 自動取引アルゴリズムを生成できるプラットフォーム
  - 画像に対するDeep Learning を金融市場関連の時系列解析等に応用
    - トレーダが投資したいチャート上の動きを提供
    - チャートを監視し、トレーダが投資したい特定のパターンを検出
  - 2016/3 iOSアプリ版をリリース

人流分析

- ABEJA
  - 店舗内のカメラ画像から人流を検出
  - 通過人数をヒートマップで表示
  - 人物の性別、年齢も推定



人が通った頻度が高い場所を赤く表示

店舗内の各場所を通った人数やその性別・年齢などをクラウドサービス上で表示

CNN：convolutional neural network

[http://techon.nikkeibp.co.jp/article/MAG/20150501/416851/]

Google AlphaGo

- Google が 2014 年に買収した、英国ベンチャー企業 DeepMind 社によって開発

- Deep Learning 技術の応用である **Deep Q-Network** (DQN) を適用
  - DQN は、Q 学習を Deep Learning で実施したもの
  - DQN に加えて、従来のモンテカルロ法探索木も併用

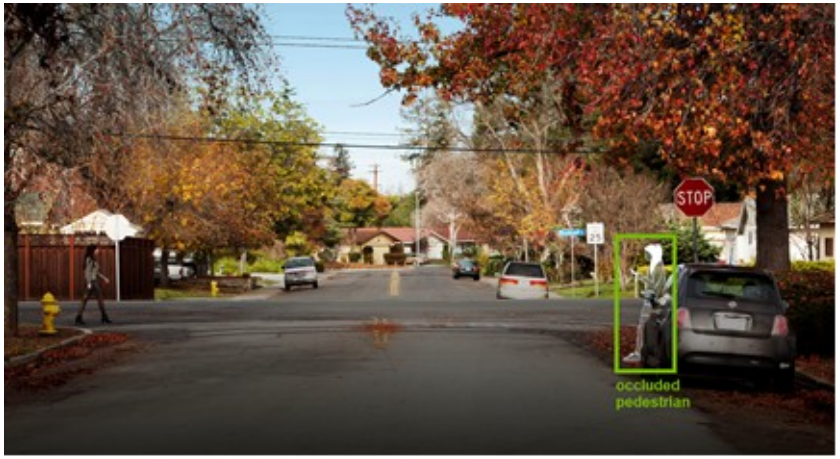
- 2016 年 3 月、世界最強棋士の一人である李世乭 九段との対戦に 4 対 1 で勝利し、一躍話題に。
  - それまでは、ゲーム探索の中で囲碁が最も難しく、プロに勝利するには数十年掛かるだろう、と言われていた。

Nature Publishing Group

自動運転

- 自動運転車の開発を各社が進めている
  - トヨタ、日産、Mercedes-Benz 、BMW、VW、GM、Audi、Tesla、、
- DeNA x ZMP でロボットタクシーを開発
  - 2020年のサービスインを目指す
- AudiはNvidiaのDrive PXを採用
  - Deep Learningを用いたComputer Visionを搭載
  - 車や歩行者などの検出が可能

出典: NVIDIA Corporation



出典: NVIDIA Corporation

出典: NVIDIA Corporation

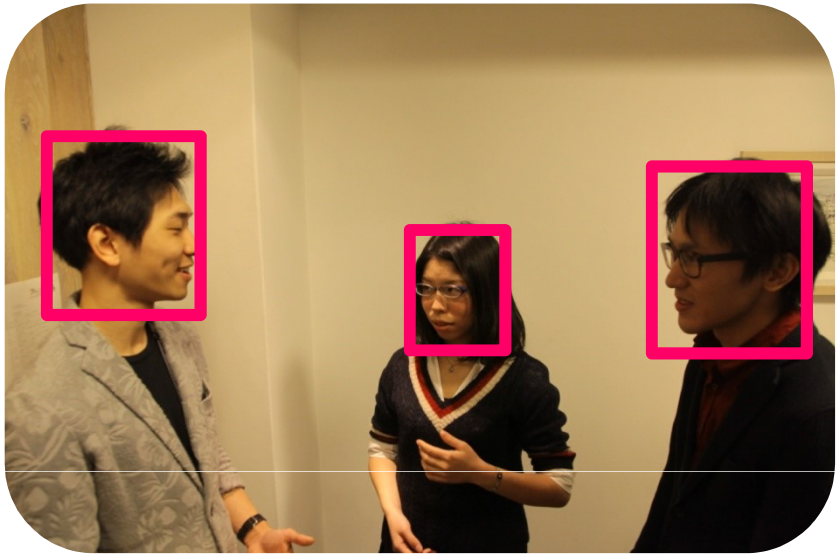
弊社事例

業種
- 自動車関連
- 通信キャリア
- 企業の研究所
- 出版
- 製造業
- 人材紹介業

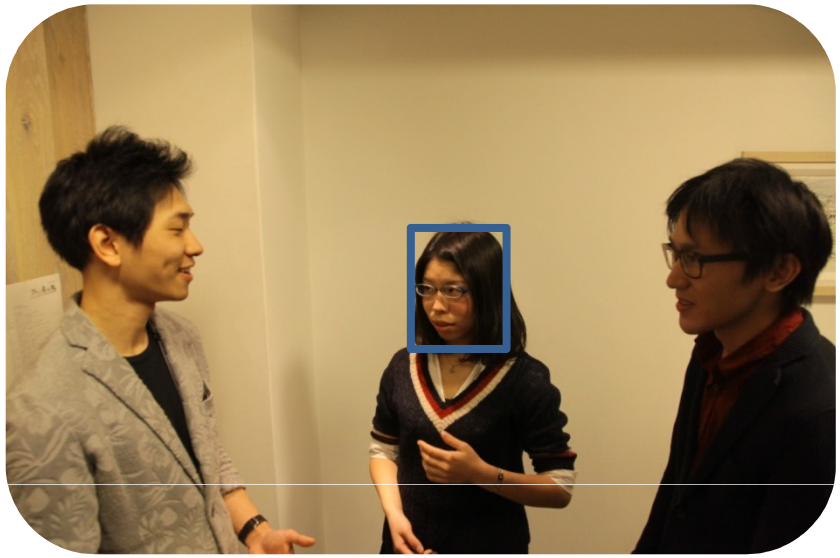
データの種類
- 画像
- 動画
- 音・音声
- 自然言語
- センサーデータ
- 上記の組み合わせ

研究・開発課題
- 判別問題
- 回帰
- 画像認識
- 類似画像検索
- 異常検知
- 逐次学習

自社エンジン 高速・高精度 リアルタイム顔検出



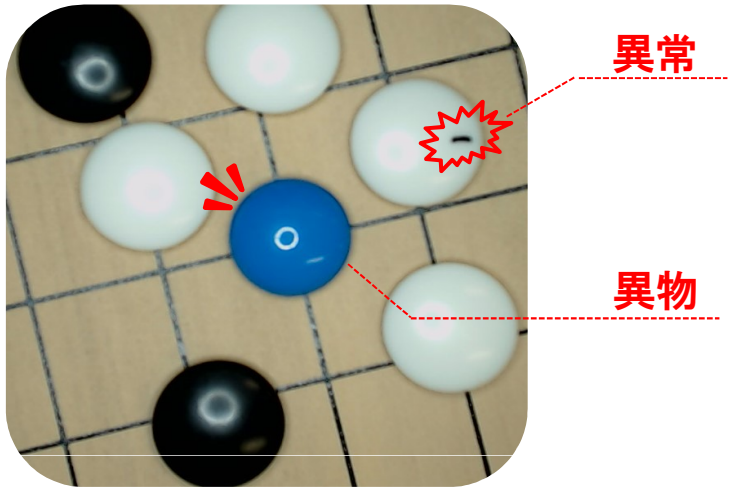
弊社手法

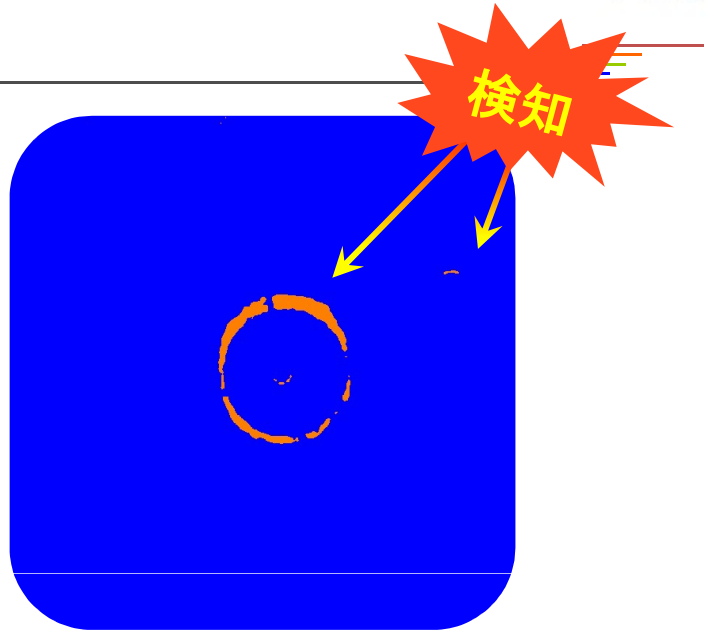


従来手法

- Deep Learningで高精度顔検出を実現
  - 横顔、低コントラスト、眼鏡ありでも検出可能
- フルカスタマイズ可能
  - 顔以外の物体検出
  - 特殊なカメラや環境に合わせたチューニング

異常検知・異物検知エンジン



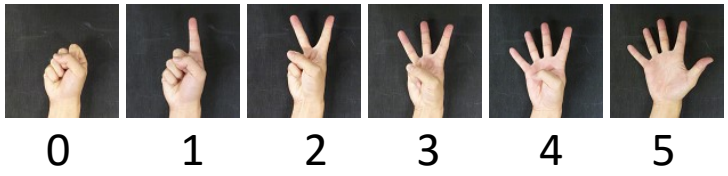


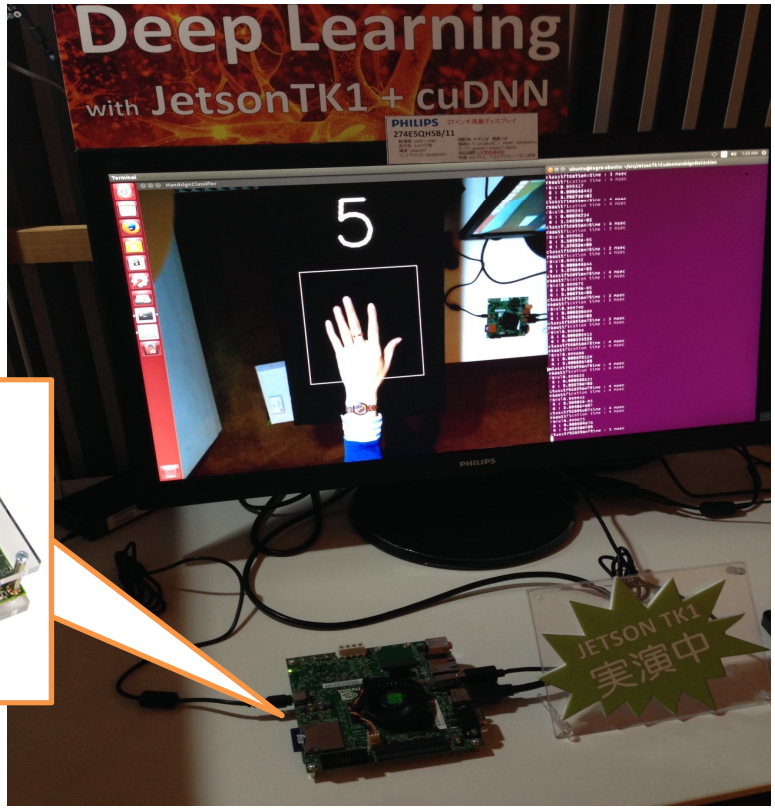
※碁石・碁盤が正常とした事例です。

- 正常データのみで学習可能
  - 正常データの数に対して、異常データは非常に少なく、バリエーションも多い
  - 「異常の学習データ」を全て揃えることは困難
- 未知の異常、異物も検知
  - 正常データ群から外れる部分を抽出し、その外れ値が「異常」であるかを判別器と組み合わせる
  - どこまでの外れ値を「異常」とするかは、課題に合わせてカスタマイズ

DLによる手形状判別

- 手形状をDLによる判別するデモ
- データ収集から学習、アプリ作成まで社内で実践してみたもの
- データ作成用専用アプリを作成
- 学習はGPUワークステーションを使用
- 判別はNVIDIA Jetson TK1で動作
- GPUでリアルタイムに判別



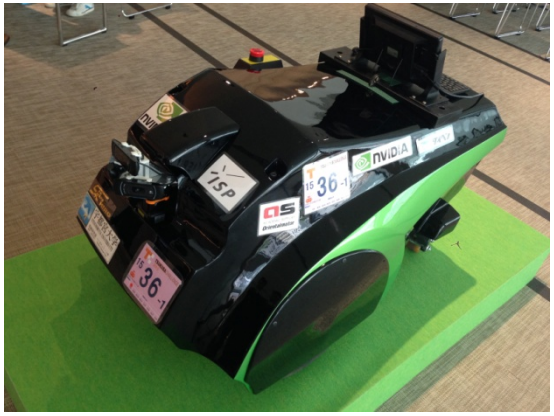


学習はGPU W/S

判別はJetsonでも可能

つくばチャレンジ

- 筑波で毎年開催される、移動ロボットに実環境を自律走行させる公開ロボット走行実験（コンテスト）
  - コースの完走を目指す
  - コース上のチェックポイント（人と看板）を検索
- ISPは宇都宮大学尾崎研究室に技術協力
  - チェックポイント（看板）検出にDeep Learningを応用



宇都宮大学尾崎研究室
MAUV

MAUV
予備走行実験の様子

今年のチェックポイント
の様子（雨天）

データ収集

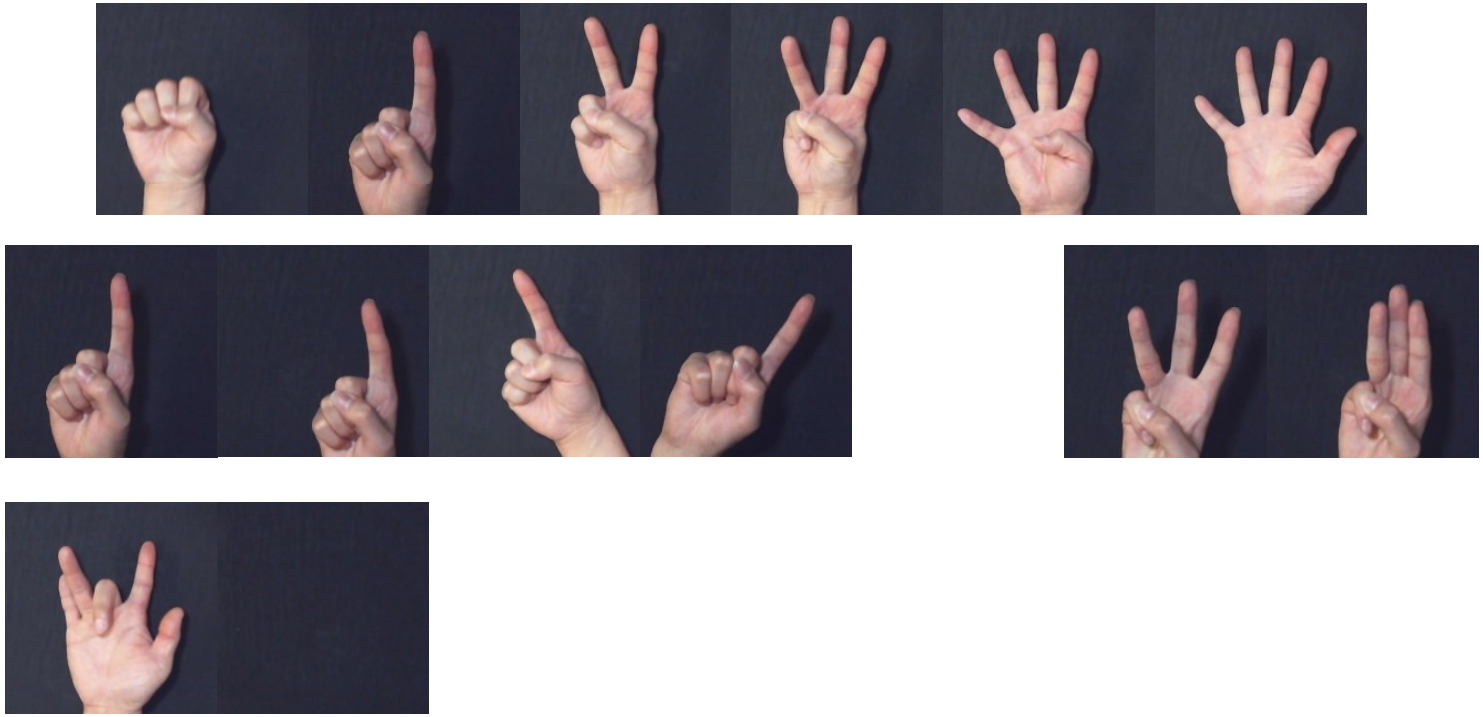
- 大量の学習画像データを画像合成技術で自動生成
  - ISP技術（クロマキー合成）を利用
  - 看板と実地の背景画像を合成し学習データを生成
  - ラベル付け（看板の有り無しと看板の位置）も自動生成

- 今後は、精度向上とJetson TK1などの省電力組み込みボード上での実行を目指し、来年のつくばチャレンジでは活躍予定！

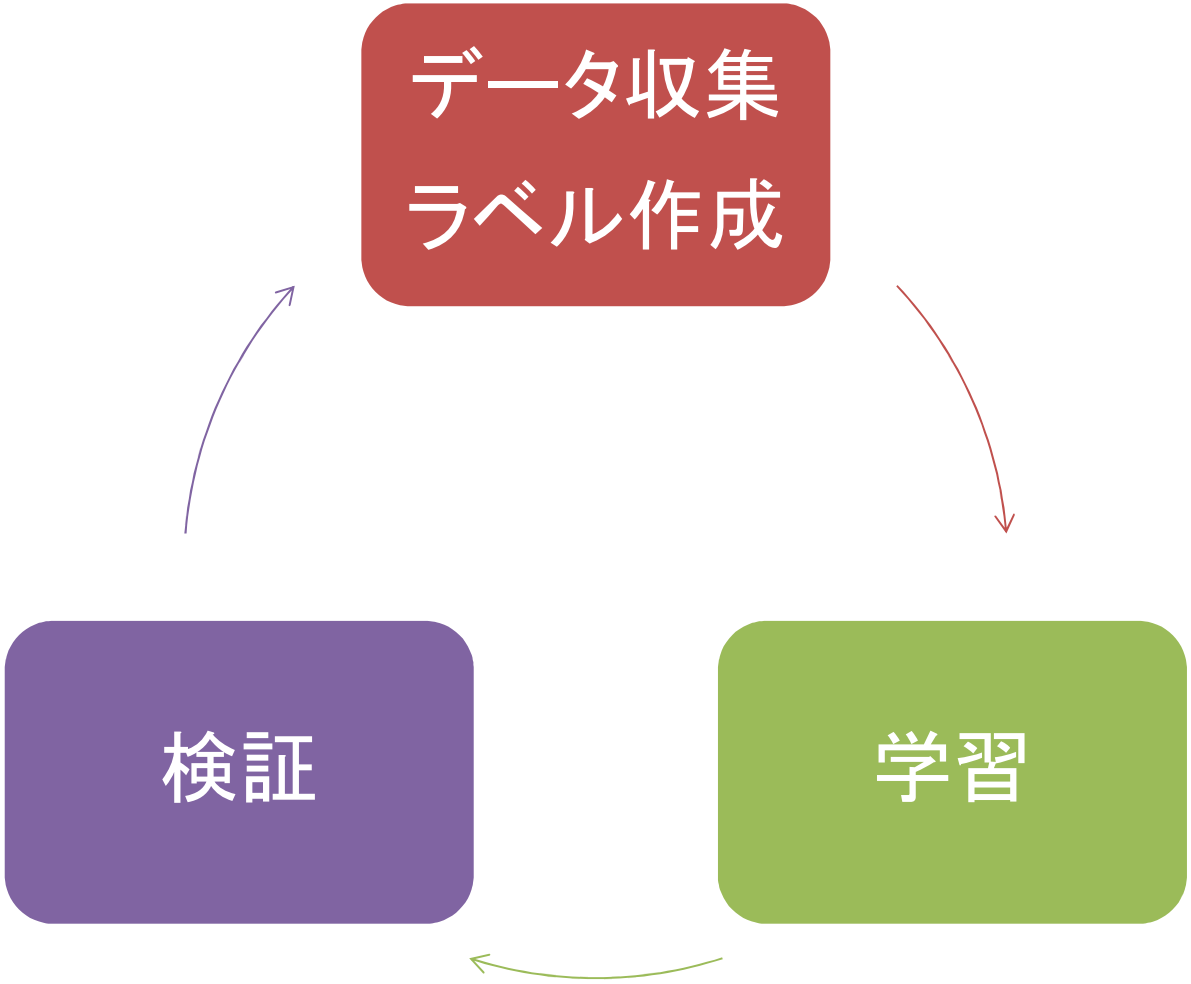
4. 開発の実際

事例1 DLによる手形状判別

- テーマ：手形状の認識

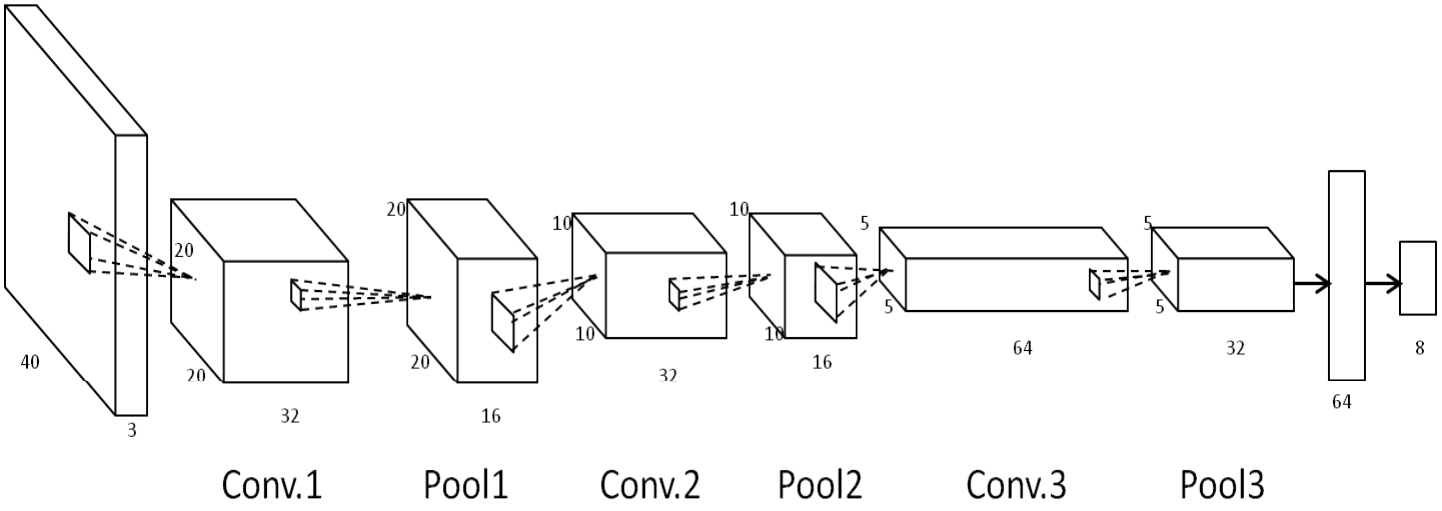


作業フロー



学習内容

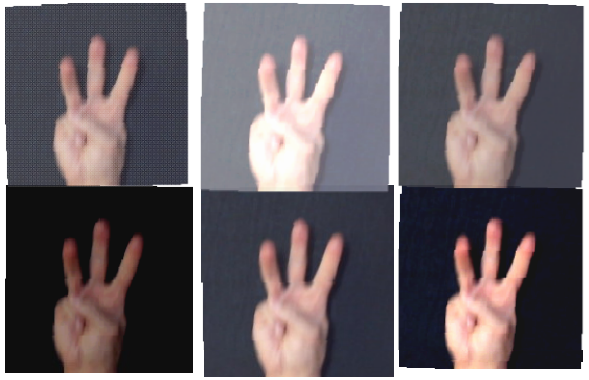
- ネットワーク構造



- CIFAR10データセットで有効なネットワークを少変更
  - 32x32 RGB、10カテゴリ
  - 今回は40x40, 8カテゴリ。32x32では手が小さくなった為。
- 学習係数は0.0001で固定
- Caffeを利用

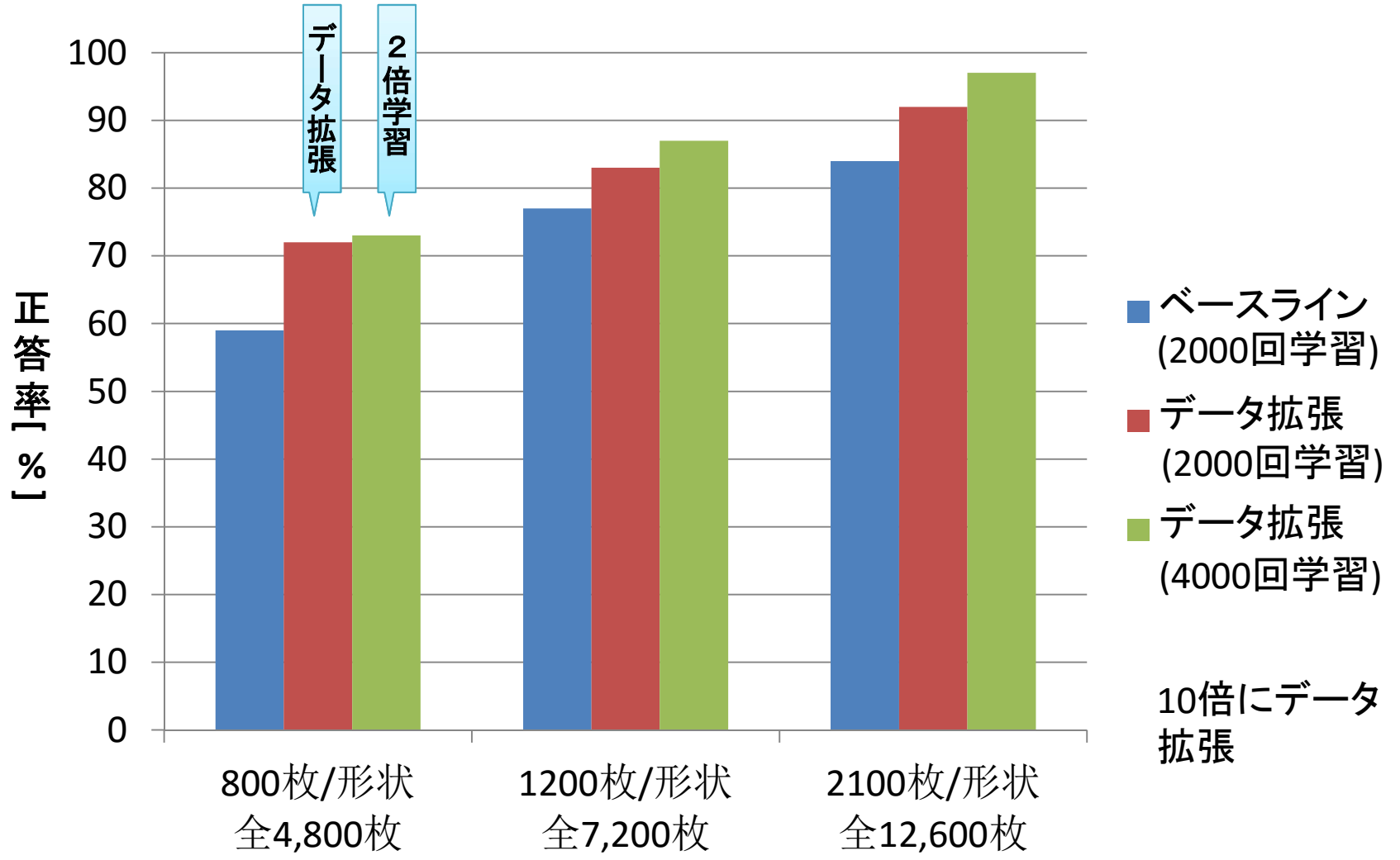
DeepLearning/CNNの性能を伸ばす為の工夫

- データ拡張
  - Data Augmentation
  - Elastic Distortion



- 前処理
  - PCA：色の偏りを利用
  - GCN：平均0、分散1
  - ZCA Whitening：ピクセル間の相関↓

実験結果 - データ拡張効果

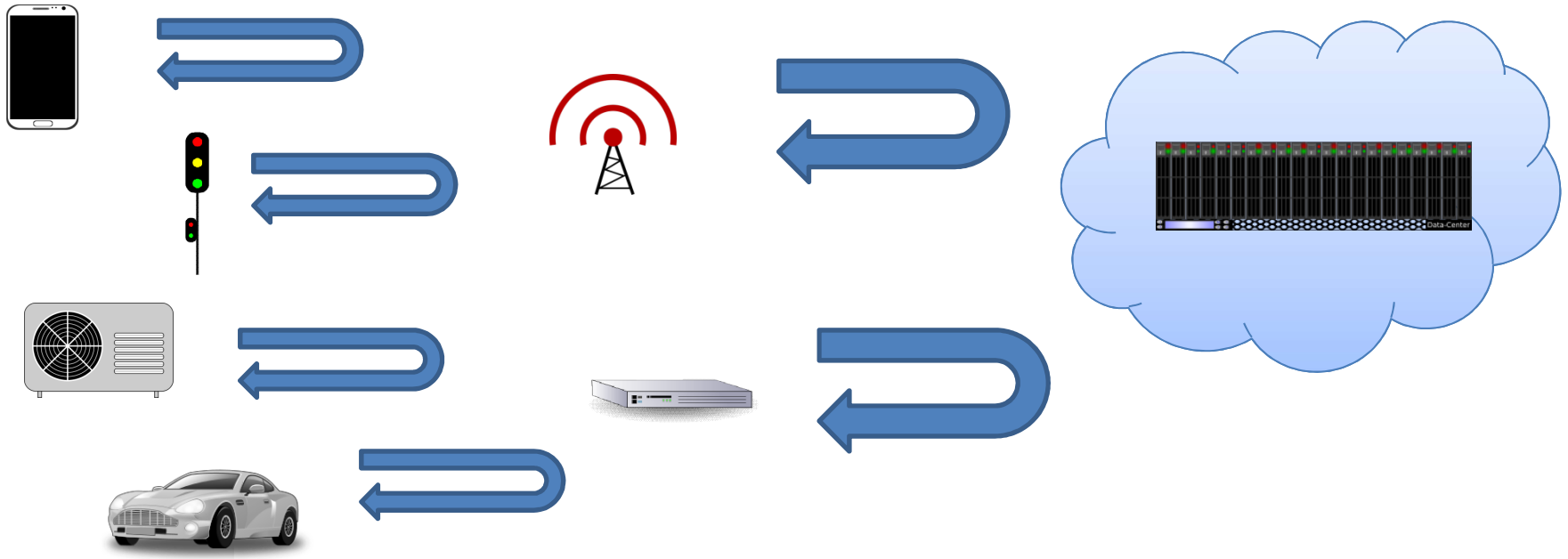


一度に100枚単位で学習

5. 組込みにおけるDLの実装

エッジコンピューティング

- ユーザに近い位置で処理を行う
  - 通信帯域の削減
  - サービスリアルタイム性
  - スタンドアロンでのサービス



Deep Learningの特性

大量の積和演算
並列処理

ヘテロジーニアス
コンピューティング

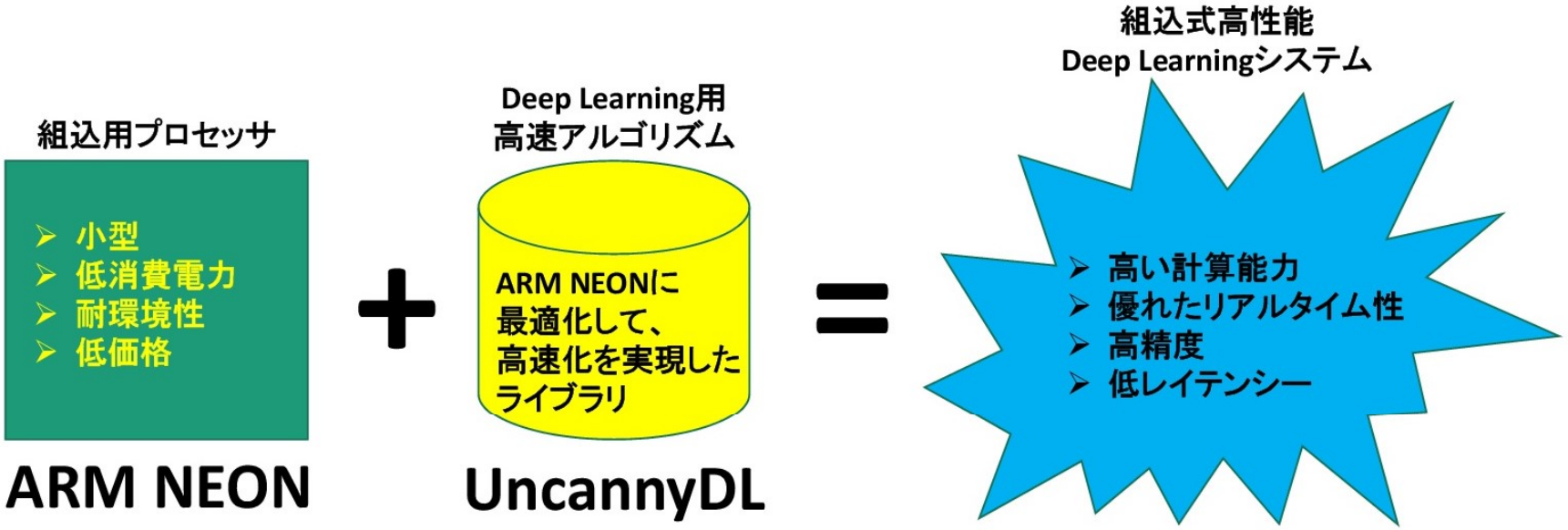
- そもそも、CPUで計算することが非効率
- 学習結果の利用だけであれば、計算量もそこそこ
- 計算量が条件に拠らず一定であり、処理時間が保証可能
- 超大量のデータを短時間で学習させる必要がなければ、エッジでも十分に処理が可能
- データ駆動型社会では、AIの目となり耳となり(センサ)、手となり足となる(ロボティクス)ようなモノが必要

さあ、我々の出番だ

ARM Neon

- ARM NEON
  - マルチメディア/信号処理向けのSIMD命令セット
- Uncanny Vision製「UncannyDL」
  - NEON命令を利用するDeep Learning (CNN)ライブラリ
  - 同社製画像処理ライブラリ UncannyCV等と組み合わせ、ADAS等に

Uncanny DL



- アルゴリズムの最適化
- C言語レベルでの最適化
- Neonの特性を活用

- アセンブラ化
- データ管理最適化
- キャッシュ最適化

Uncanny CV

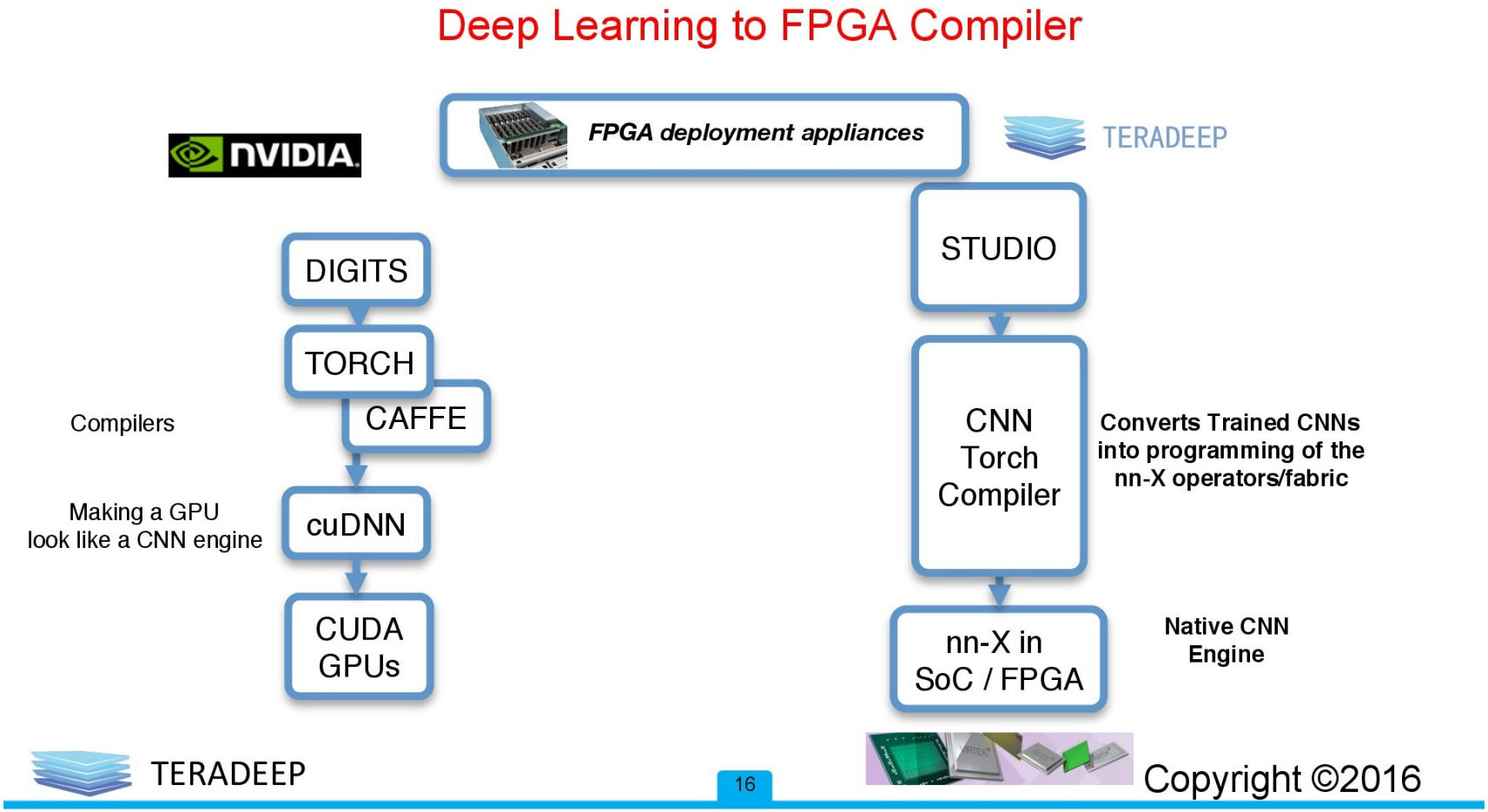
OpenCVとの比較で2～20倍の性能（1GHz Cortex A9 で測定）

Algorithm アルゴリズム	Throughput スループット (Megapixels per sec) （メガピクセル/秒）	Acceleration 倍速度 (OpenCVとの比較)
Canny Edge detection　キャニーエッジ検出	25.0	3 x
ORB (1500 keypoints)	3.7	5x
Convolution 5x5 畳み込みフィルター5x5	96	22x
Erosion/Dilation 収縮/膨張	153	6.5x
Integral Image インテグラルイメージング	96	2.4 x
Harris Corner ハリスコーナ検出	15.7	6.5x
Stereo Depth Computation	(Parameter dependent)	6.1x
Face Detection (LBP Cascade)	(Parameter dependent)	3.5x
Connected Components 連結成分抽出	(Image Dependent)	1.7 x
HOG Pedestrian Detection 歩行者検出	1.7 (on Cortex-15)	9x

NVIDIA CUDA (GPGPU)

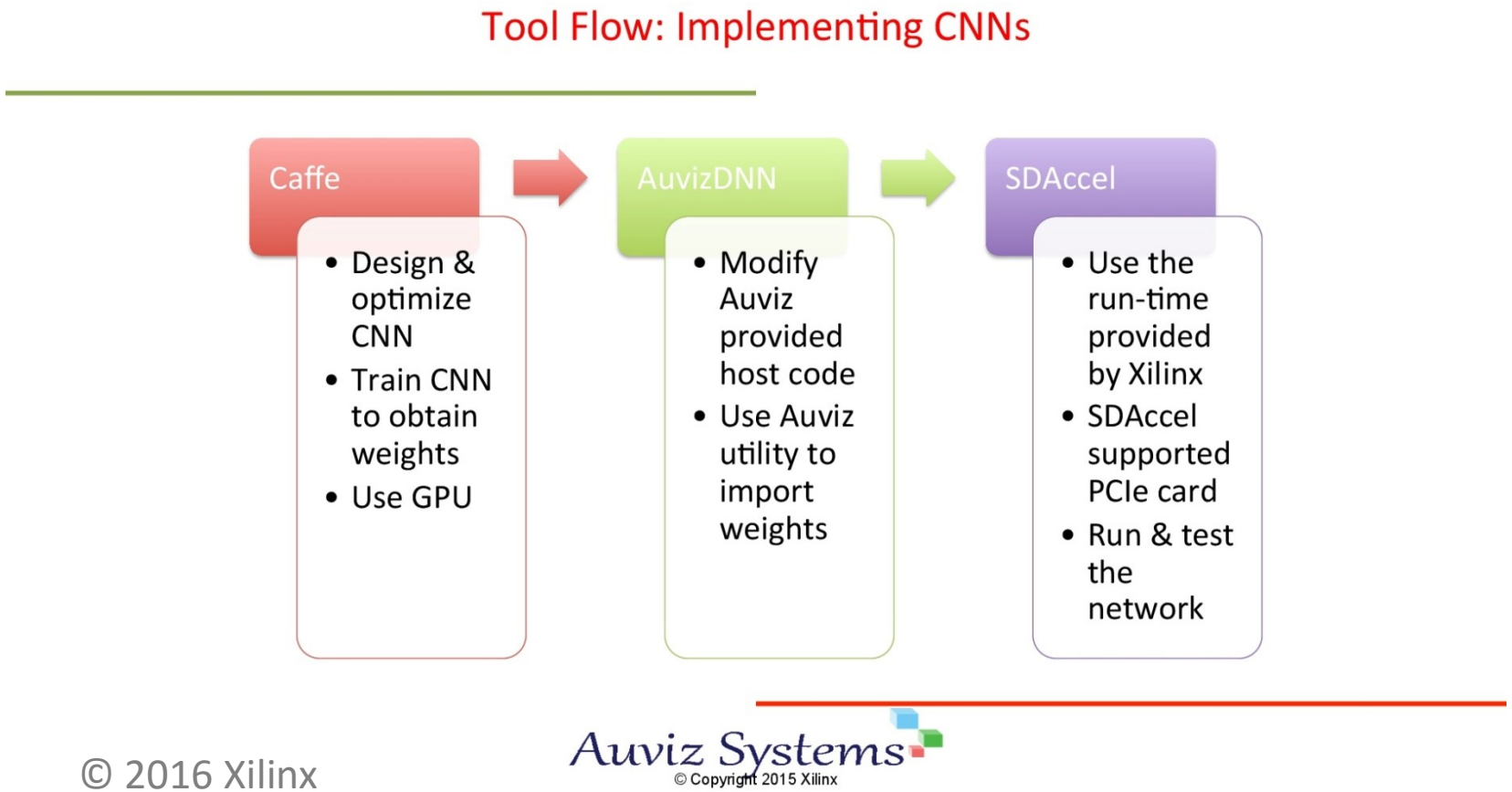
- NVIDIA製GPGPU向け開発環境
  - cuDNN等のライブラリを利用
  - 生で処理を書くことも可能
  - CUDAに対応したミドルウェア(Caffe, Chainer, Tensorflow等)が利用可能
- 評価キット
  - Jetson TX1
    - Tegra X1 (ARM A57 x 4 + A53 x 4 + CUDA CORE x 256)
  - Jetson TK1
    - Tegra K1 (ARM A15 + CUDA CORE x 192)
- 自動運転開発プラットフォーム
  - Drive PX
    - Tegra X1 x 2

FPGAの利用

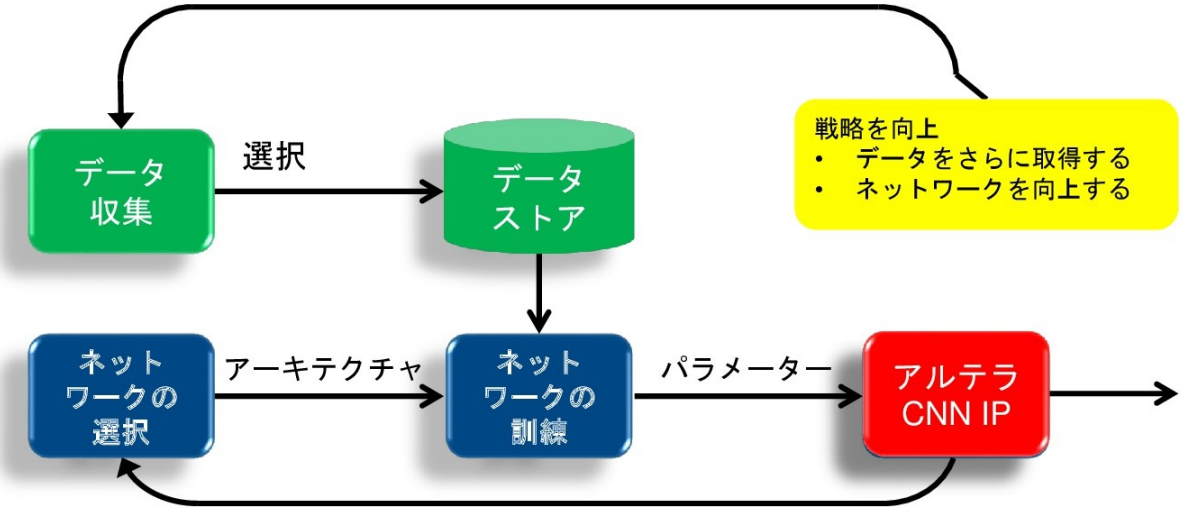


Xilinx

- Auviz Systems製 IP Core「Auviz DNN」



例:CNN IP を使用した設計フロー



ネットワークの選択
- フレームワークの使用（例: Caffé、Torch）
- 経験に基づいて選択、または実行エンジンを制限する

ネットワークのトレーニング
- HPC の作業負荷
- データをあらかじめ選ぶ必要がある
- 処理に数週間から数か月かかる

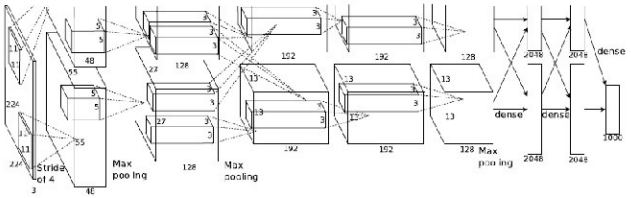
実行エンジン
- ニューラル・ネットワークの実装
- 柔軟性、パフォーマンス、およびパワーによって選択肢が決まる

22 © 2016 Altera—Public

ALTERA now part of Intel

Case Study: Image Classification

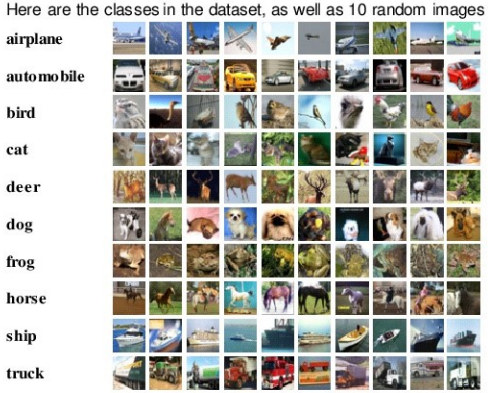
- Deep Learning Algorithm
  - Convolutional Neural Networki
  - Based on Hinton's CNN
- Early Results on Stratix V
  - 2X Perf./Power vs. gpgpu
    - despite soft floating point
  - 8+ simultaneous kernels
    - vs. 2 on gpgpu
  - Exploiting OpenCL channels
    - between kernels
- A10 Expectations
  - Hard floating point
  - Better density and frequency
  - ~ 4X performance/watt v SV



Hinton's CNN Algorithm

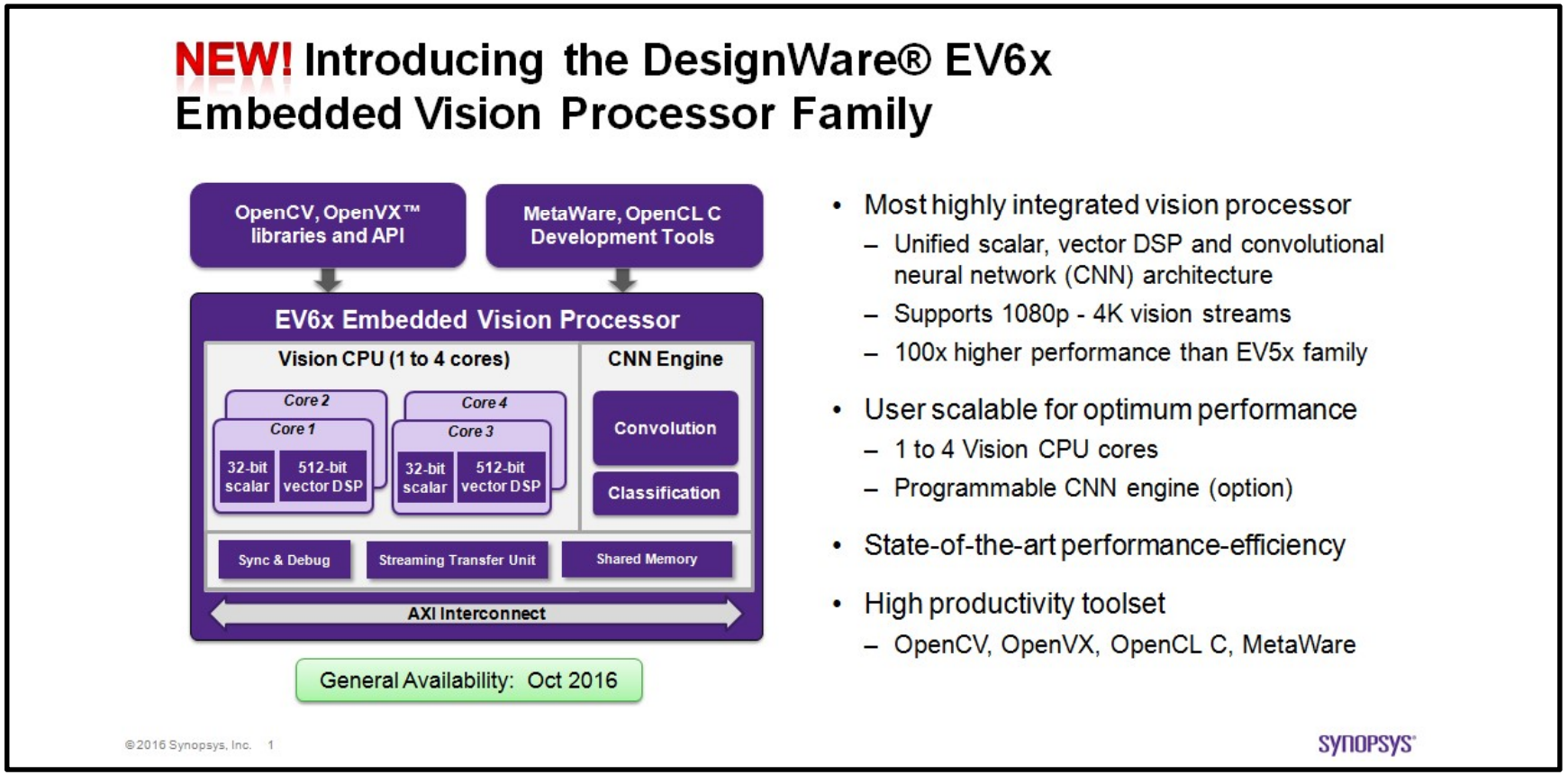
The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Here are the classes in the dataset, as well as 10 random images from each:



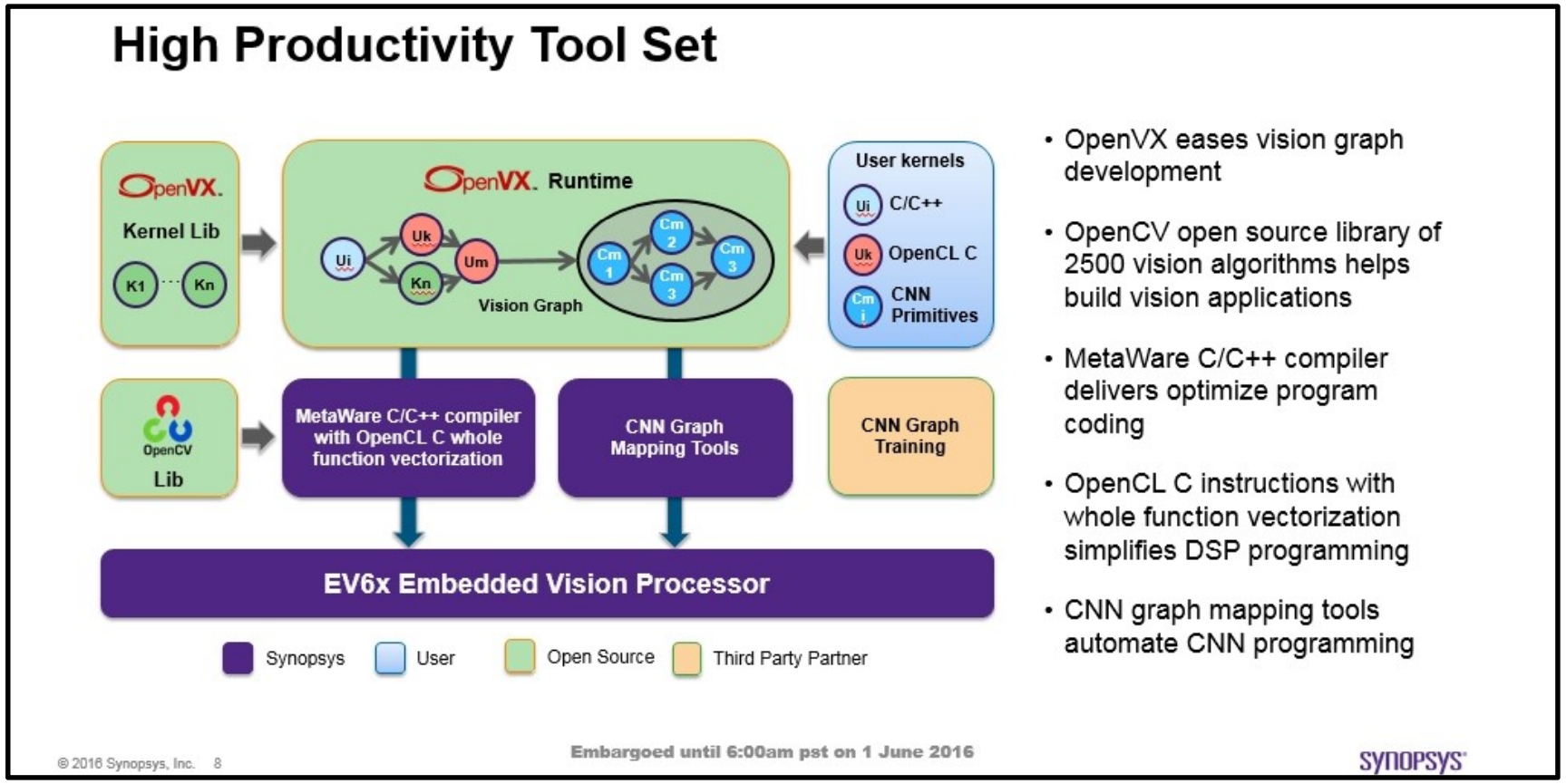
Synopsys EV6x（DL専用エンジン）

- 512bit DSP を有するハイパワーARC コアとプログラマラブルな CNN Engine の組み合わせ
- ワークステーションで学習したネットワークを、CNN Graph Mapping Tool でマッピング可能

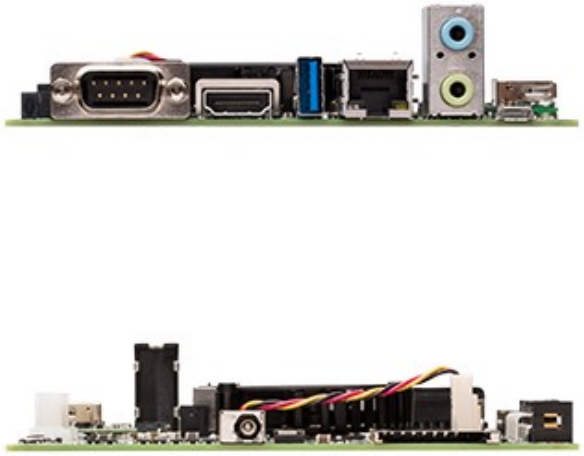


Synopsys EV6x（DL専用エンジン）

- ADASおよび監視カメラシステムなどを想定、4K画像に対応可能
- OpenCV, OpenVX, OpenCL, MetaWare による開発



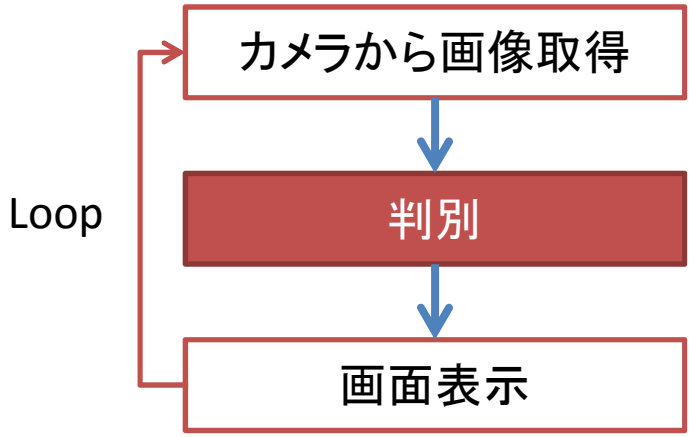
弊社事例1 NVIDIA Jetson TK1



- NVIDIA Tegra K1 SOC 搭載 評価ボード
- 先ほどの手形状判別をJetson TK1上に構築

ISP + Jetson TK1 + cuDNN v2

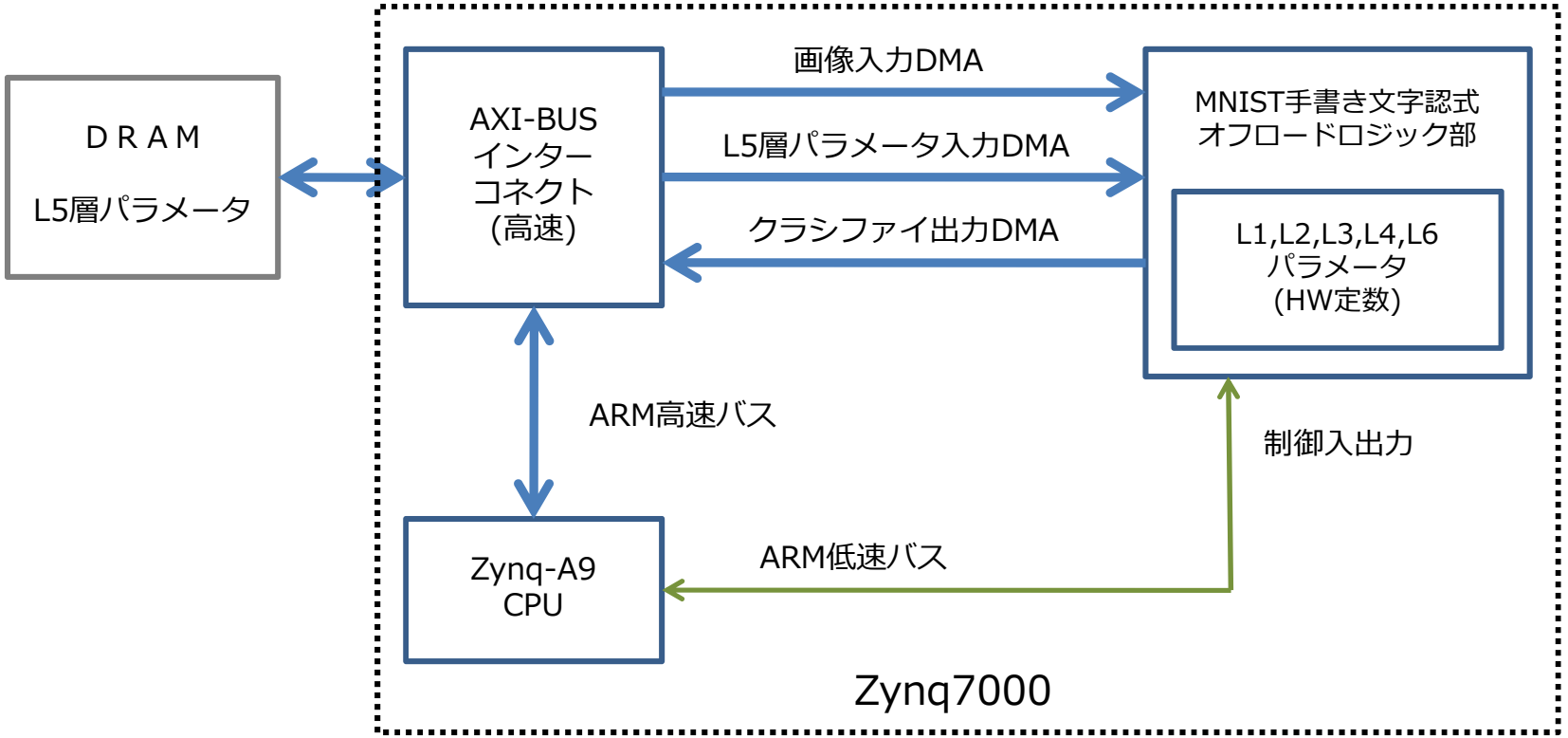
- caffeをそのまま移植(Python / C++ / cuDNNv1)
  ⇒ 判別に23～25ms...
- caffeのpython部分を削除
  ⇒ 判別に20ms ～ 30ms
- cuDNNv2ベースのISP 独自実装
  ⇒ 判別に5ms程度
  - デバイス制御回りの工夫



弊社事例2 Xilinx Zynq

- tiny-cnn を、Xilinx Zynq7020上に実装
  - https://github.com/nyanp/tiny-cnn で公開されているC++実装
  - サンプルの、LeNetによるMNIST手書き文字認識をFPGA化
    - LeNetはConvolition 3層、Sampling 2層、Fully Connected 1層の小さなネットワーク
    - 入力する文字データは、28画素x28画素のグレイスケール画像
    - ネットワークパラメータ(≒5万)は Zynq7020 のBRAM に全部載った
- 高位合成ツール(SDSoC)で開発
  - C++のテンプレートメタプログラミングを利用しているため、任意サイズのネットワークに対応したHDL生成が可能
- レイヤー毎に固定小数点の有効桁数をチューニング
  - リソース使用量の削減に寄与

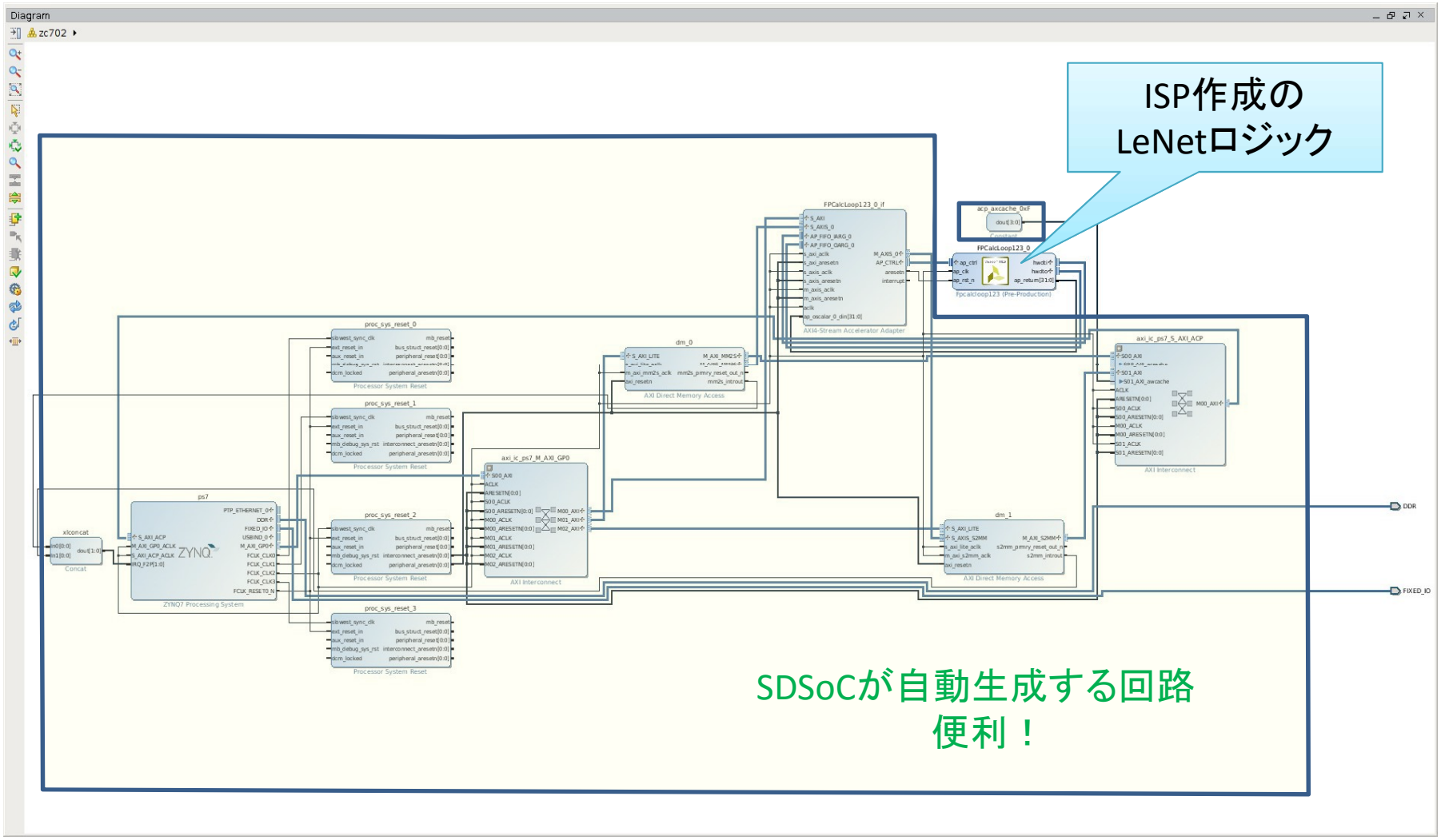
LeNetロジック配置図



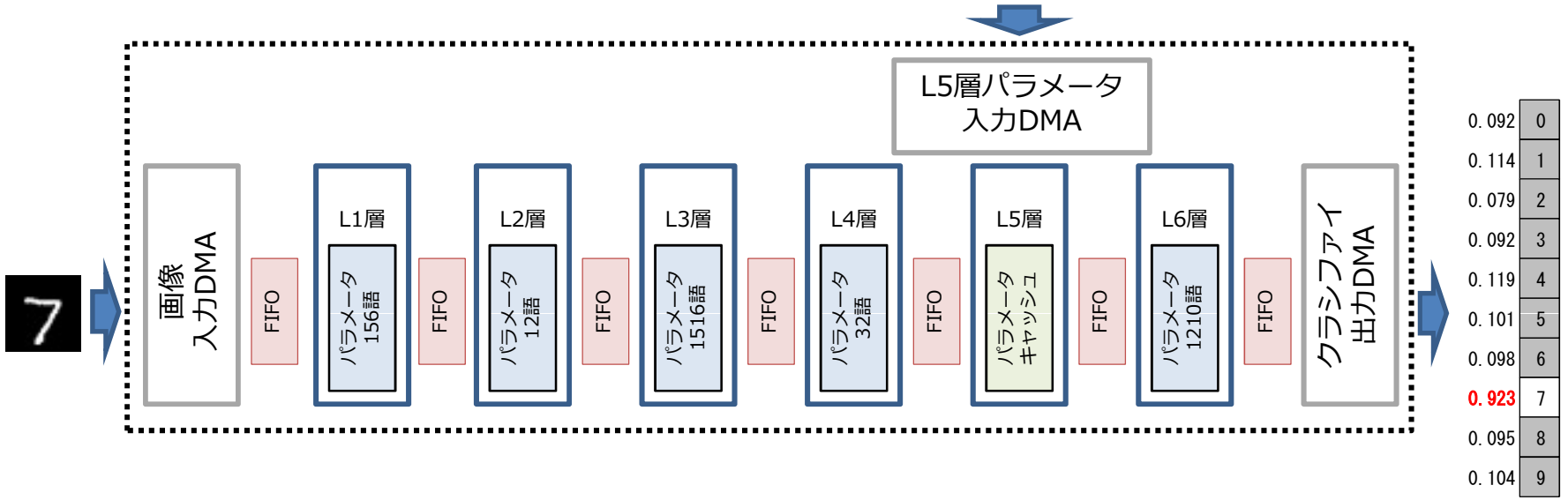
MNIST手書き文字認識ではL5層のパラメータが全パラメータの94%を占めるため、
L5層のパラメータを外部に配置し、それ以外のパラメータはHWに直接組み込んでいる。

SDSoCで開発を行った場合は、オフロードロジック(関数)に必要なパラメータのDMA及び、
ロジックの制御入出力は自動で生成される。

SDSoCでのブロック図



LeNet ロジックブロック拡大

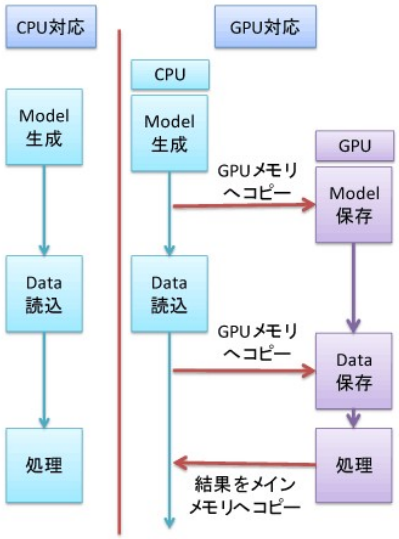


- L1,L3,L5層は畳み込み層で処理が重く、層別パイプライン化でのインターバルの決定要素になる。
- L1,L3,L5各層で処理量が違うので、並列度を変えて各層のインターバルを合わせている。
- 各層のネットワークパラメータ及び並列度パラメータはC++テンプレートのパラメータで変更可能。

エッジ側 開発上の注意点

- HW化を念頭に置いたニューラルネットワーク設計が必要
  - あまり大きなネットワークは利用できない
- 問題設定が重要
  - ネットワークの表現力に収まるような問題設定とする
  - 複雑な処理が必要であれば、クラウド側との分業で
  - ヒューリスティックな工夫も大切

技術公開サイト「技ラボ」

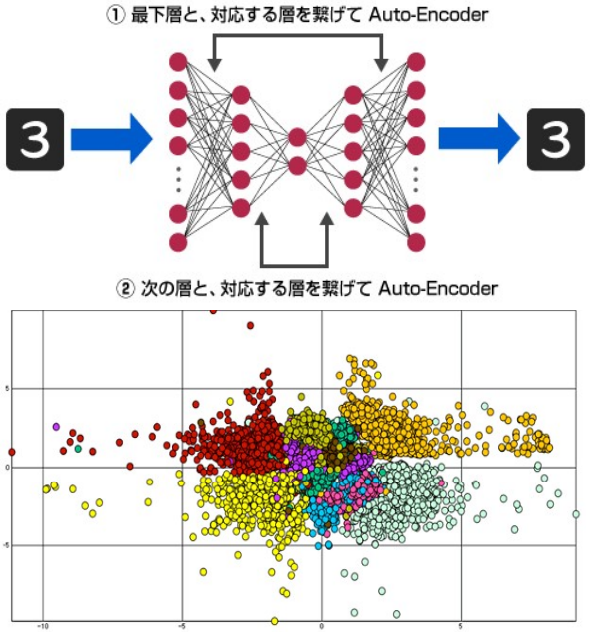


・XchainerをGPU対応させてみた

技ラボ
wazalabo.com

様々な技術情報を公開しています

http://wazalabo.com/



・ChainerでStacked Auto-Encoderを試してみた

株式会社 システム計画研究所／ISP
Research Institute of Systems Planning, Inc.

TEL:03-5489-0232
E-Mail:ai-contact@isp.co.jp

「ひと」だからできること、
技術者として担うにたる仕事をしたい。
求めるものは「創造」。
そのためには、広く浅くより
「より深く、先端へ」

ISPの技術情報と関連製品の紹介を行っています。

技ラボ
wazalabo.com

http://wazalabo.com/