Solution Approach

in Integration of AI Engine into AGL

Japan Technical Jamboree 64

March 2nd, 2018

NTT DATA MSE Corporation
Hiroto Imamura
Tomonari Okuno

NTT DATA MSE Corporation

Who are we?

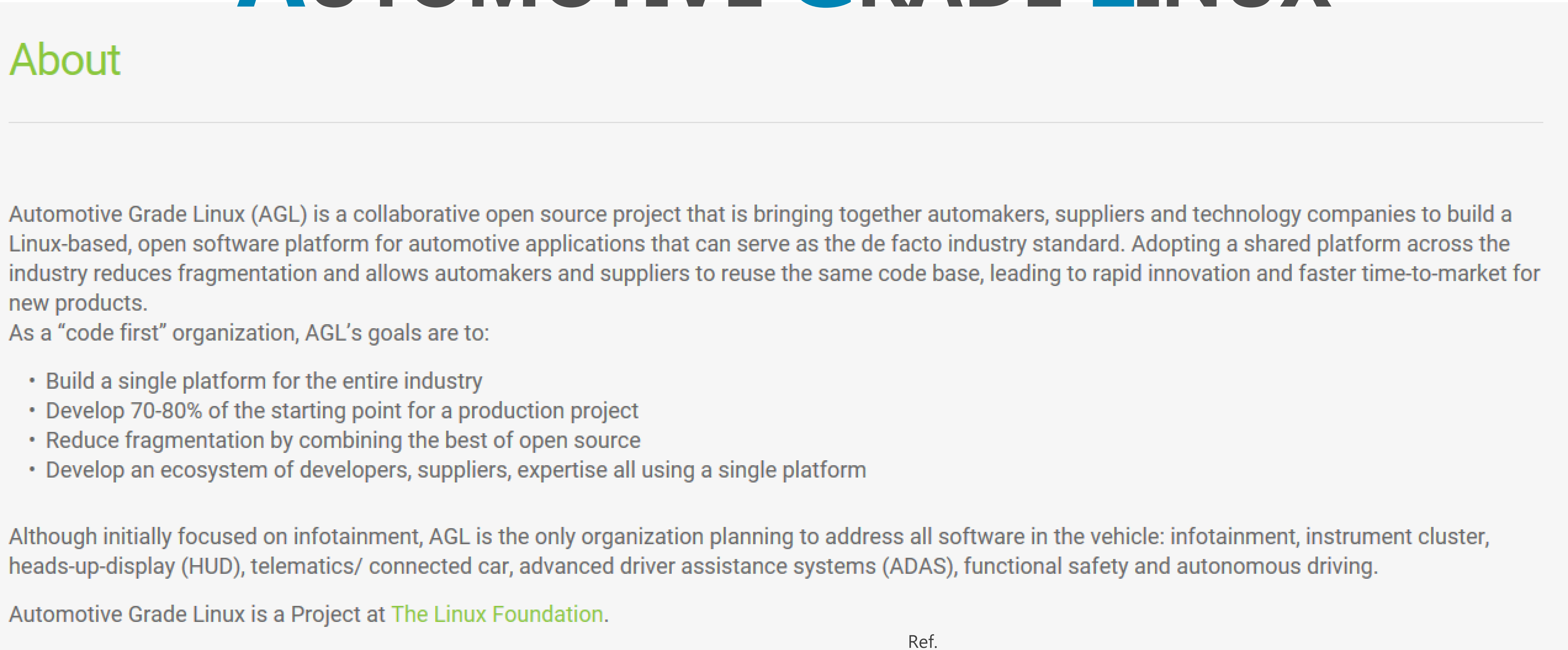
NTT DATA MSE Corporation

Name	**Hiroto Imamura**
Position	**Manager**
Carrier	**- Leader of OSS Collaboration related activities in NTT DATA MSE** **- Linux-based embedded devices** **- Architecture design / System debugging / Performance optimization / Security**

Name	**Tomonari Okuno**
Position	**Deputy Manager**
Carrier	**- Lead Architect of R&D projects in NTT DATA MSE** **- Linux-based embedded devices** **- Performance optimization / OSS Licenses**

What is AGL?

AUTOMOTIVE GRADE LINUX



About

Automotive Grade Linux (AGL) is a collaborative open source project that is bringing together automakers, suppliers and technology companies to build a Linux-based, open software platform for automotive applications that can serve as the de facto industry standard. Adopting a shared platform across the industry reduces fragmentation and allows automakers and suppliers to reuse the same code base, leading to rapid innovation and faster time-to-market for new products.
As a “code first” organization, AGL’s goals are to:

- Build a single platform for the entire industry
- Develop 70-80% of the starting point for a production project
- Reduce fragmentation by combining the best of open source
- Develop an ecosystem of developers, suppliers, expertise all using a single platform

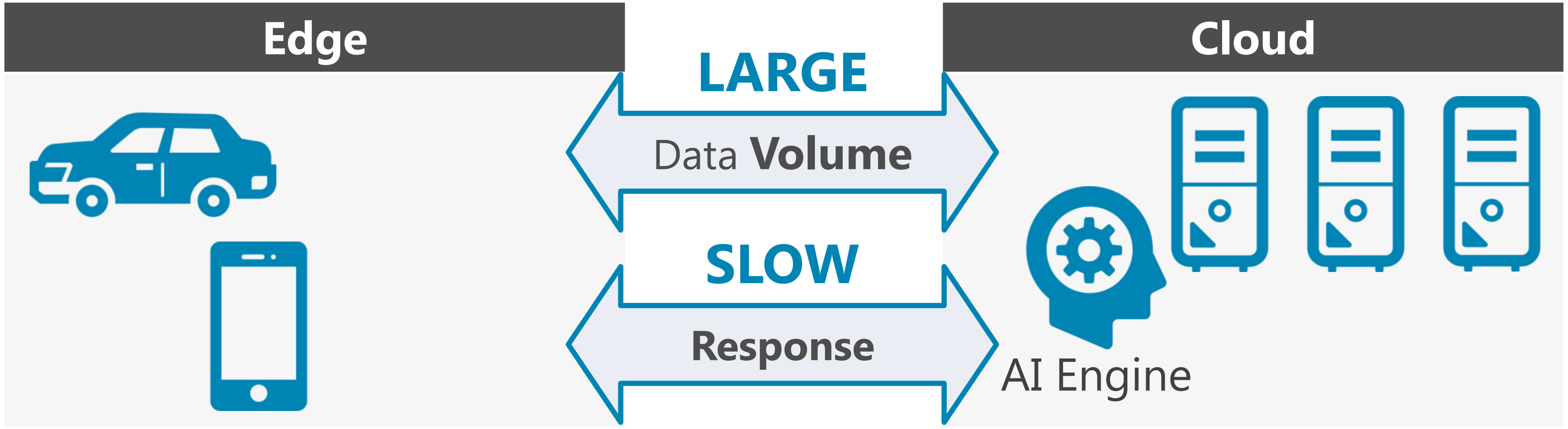
Although initially focused on infotainment, AGL is the only organization planning to address all software in the vehicle: infotainment, instrument cluster, heads-up-display (HUD), telematics/ connected car, advanced driver assistance systems (ADAS), functional safety and autonomous driving.

Automotive Grade Linux is a Project at The Linux Foundation.

Ref.
- https://www.automotivelinux.org/about
- https://www.automotivelinux.org

Why Edge AI ?

So far



Edge

Cloud

LARGE

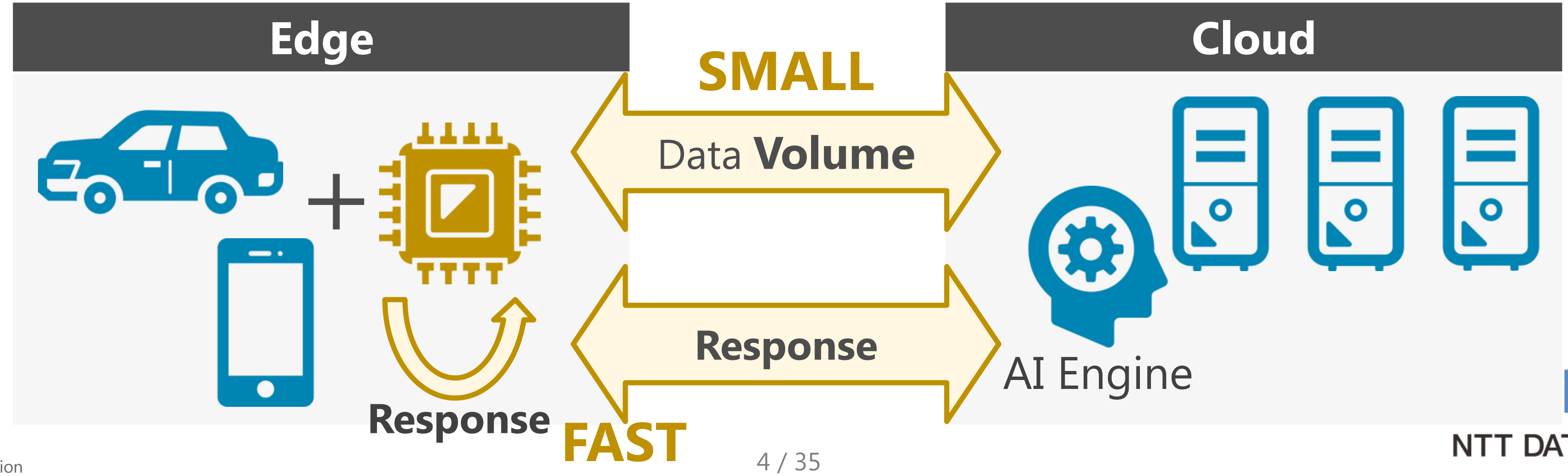
Data **Volume**

SLOW

Response

AI Engine

Near future



Edge

Cloud

SMALL

Data **Volume**

Response

+

Response

FAST

AI Engine

Edge AI is spotlighted

More and more Edge AI solutions have been announced

Apr. 2017	**e-AI** by Renesas Electronics
Jun. 2017	**Neural Network Libraries** by Sony
Jun. 2017	**Embedded Learning Library** by Microsoft
Jul. 2017	**TensorFlow Lite** by Google
Jul. 2017	**revision** by Xilinx
Sep. 2017	**Kirin 970** by HUAWEI
Jan. 2018	**Deep Learning Accelerator Card** by PFU

Our motivation

1. OSS / Rich development environment **Easy to try**
2. Few people addressed **Good chance to appeal our technical capabilities**
3. AGL Member **Make collaboration**

Collaboration

supported by Sony members

NTT DATA MSE Corporation

Feedback at CES 2018

Valuable opinions, Valuable discussions

NTT DATA MSE Corporation

Technical Explanation

Introduction

■ Our team have started AI related activities from October 2017

- There are AI related news almost everyday
- Our interest:
  - ✓ How is the performance of AI on edge devices?
  - ✓ What do we need to learn in order to realize AI on edge devices?
- First Step: Let’s use an AI engine on edge devices
- Implemented a demo system “Handwritten Digit Recognition App”

■ In this presentation

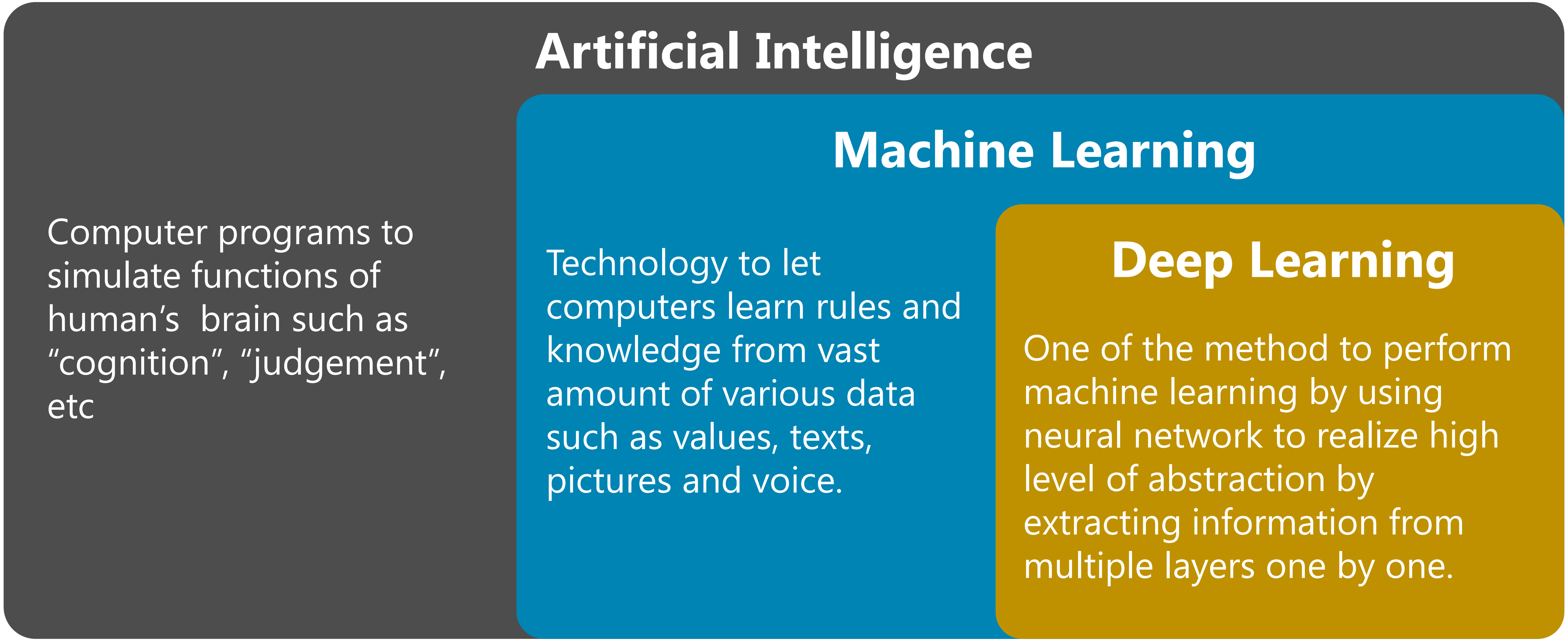
- Overview of Machine Learning on edge devices
- How we implemented the “Handwritten Digit Recognition App” on AGL

NTT DATA MSE Corporation

What is AI?

AI, Machine Learning, Deep Learning …

What are the relations of these terms?



Artificial Intelligence

Computer programs to simulate functions of human’s brain such as “cognition”, “judgement”, etc

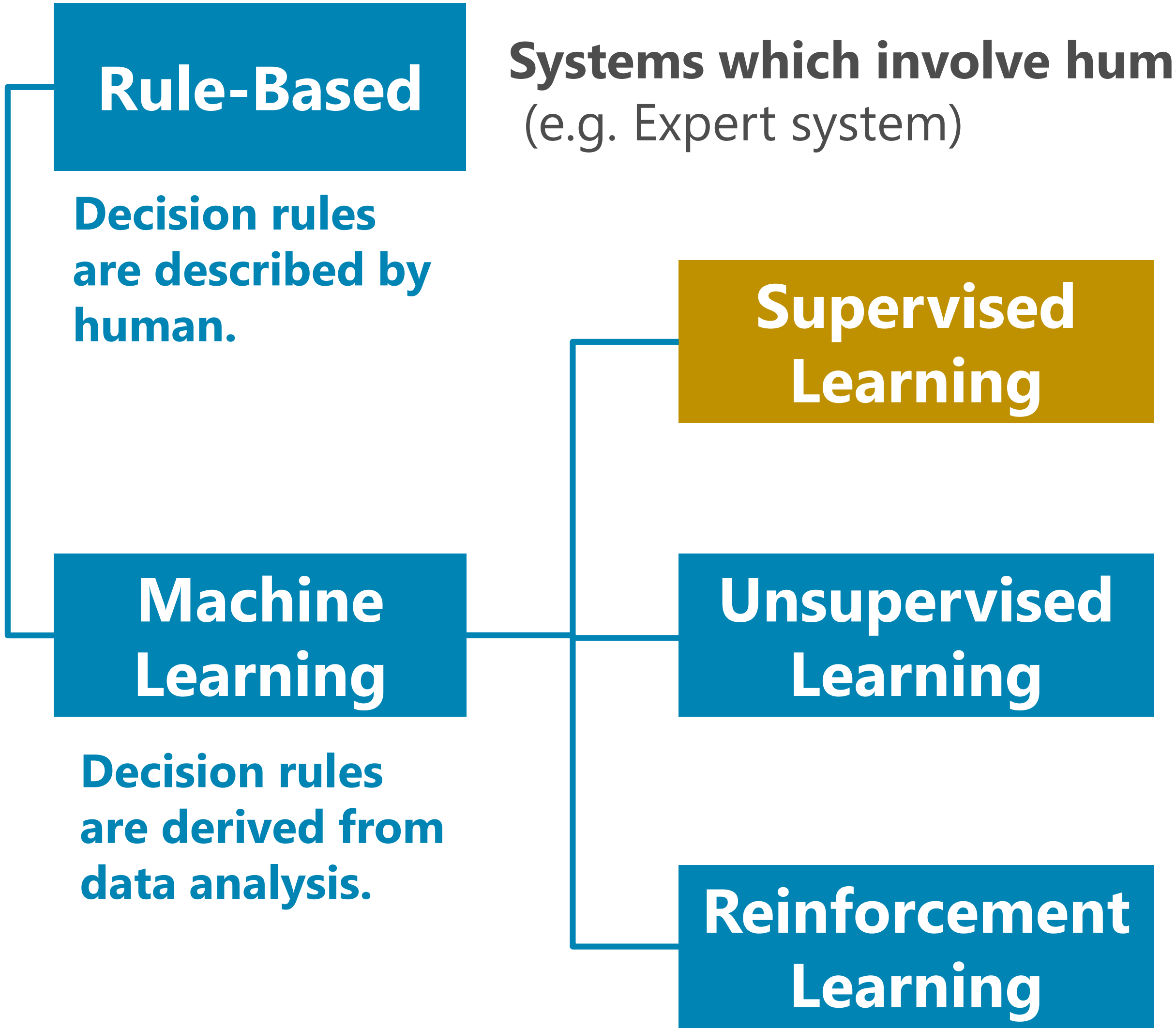
Machine Learning

Technology to let computers learn rules and knowledge from vast amount of various data such as values, texts, pictures and voice.

Deep Learning

One of the method to perform machine learning by using neural network to realize high level of abstraction by extracting information from multiple layers one by one.

Types of AI



Rule-Based

Systems which involve human created rule sets.
(e.g. Expert system)

Decision rules are described by human.

Machine Learning

Decision rules are derived from data analysis.

Supervised Learning

Systems which learn from given labeled training data(=Answers are given from a teacher).
(e.g. Voice/Image Recognition, Stock Price Prediction)

Unsupervised Learning

Systems which learn from given unlabeled training data(=Answers are not given).
(e.g. Automatic data classification, Anomaly detection)

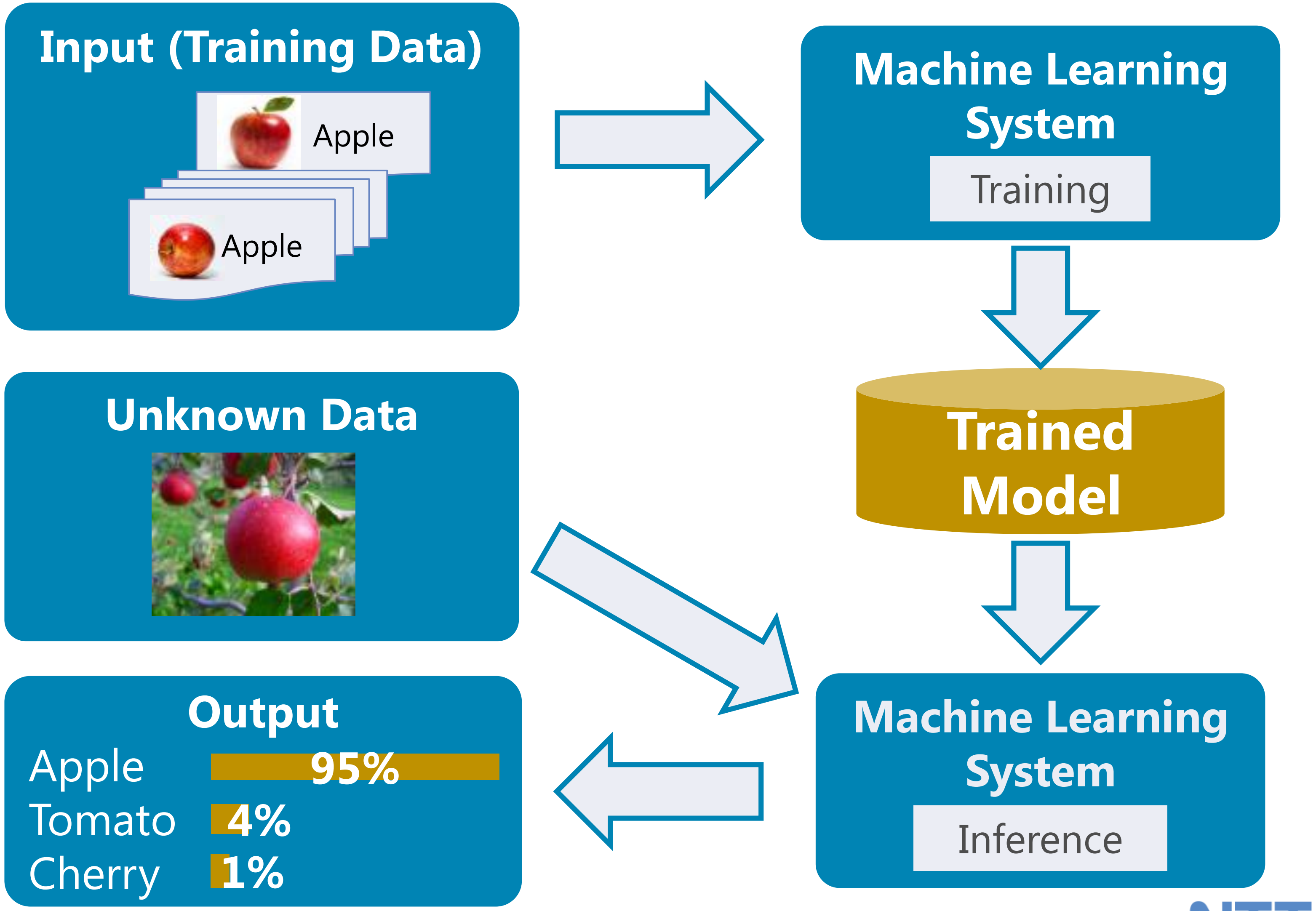
Reinforcement Learning

Systems which learn in an environment where only the final result evaluations are given.
(e.g. AI-Shogi, AI-Go)

Machine Learning (Supervised learning)

Training

Prediction



Input (Training Data)

Apple

Apple

Machine Learning System

Training

Trained Model

Unknown Data

Machine Learning System

Inference

Output

Apple	95%
Tomato	4%
Cherry	1%

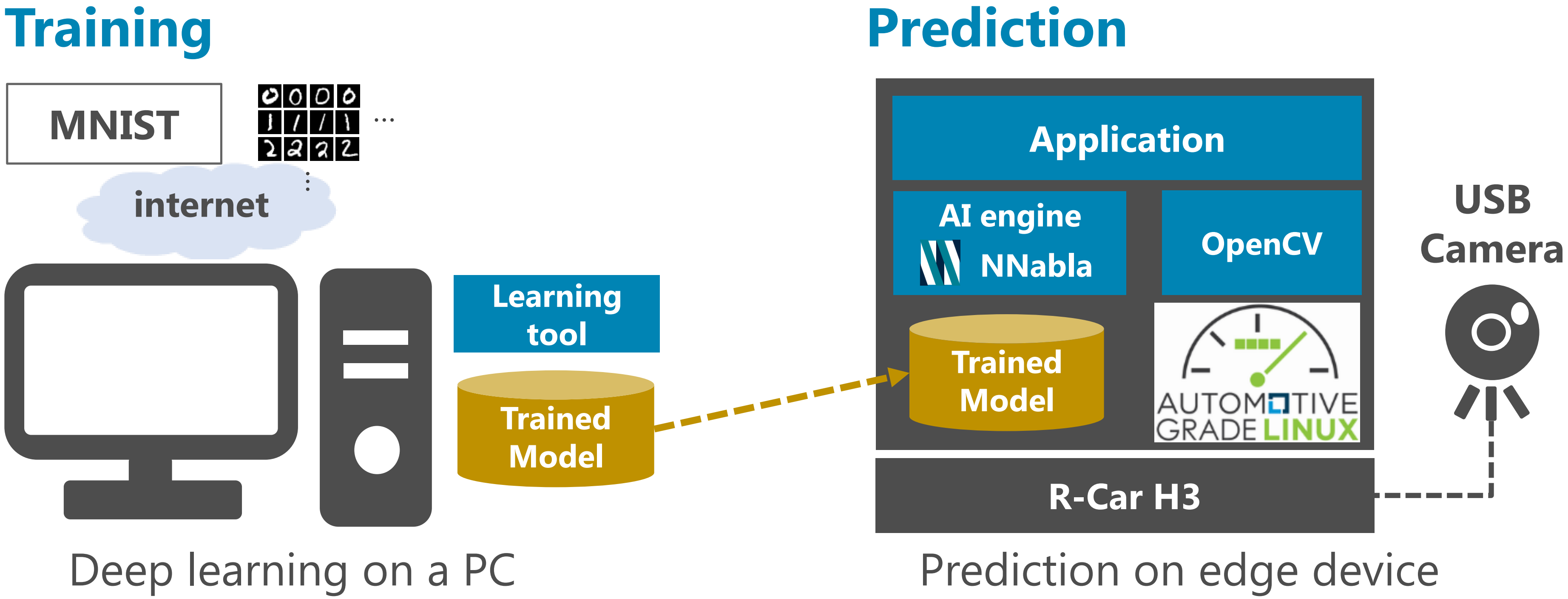
NTT DATA

NTT DATA MSE Corporation

Demo System:
Handwritten Digit Recognition App

System Overview

- Recognizes handwritten digits from an image captured with an USB Camera
- Deployed a pre-trained model to an edge device



Software version

AGL: EE 5.0.0
NNabla: v0.9.6
OpenCV: v3.2

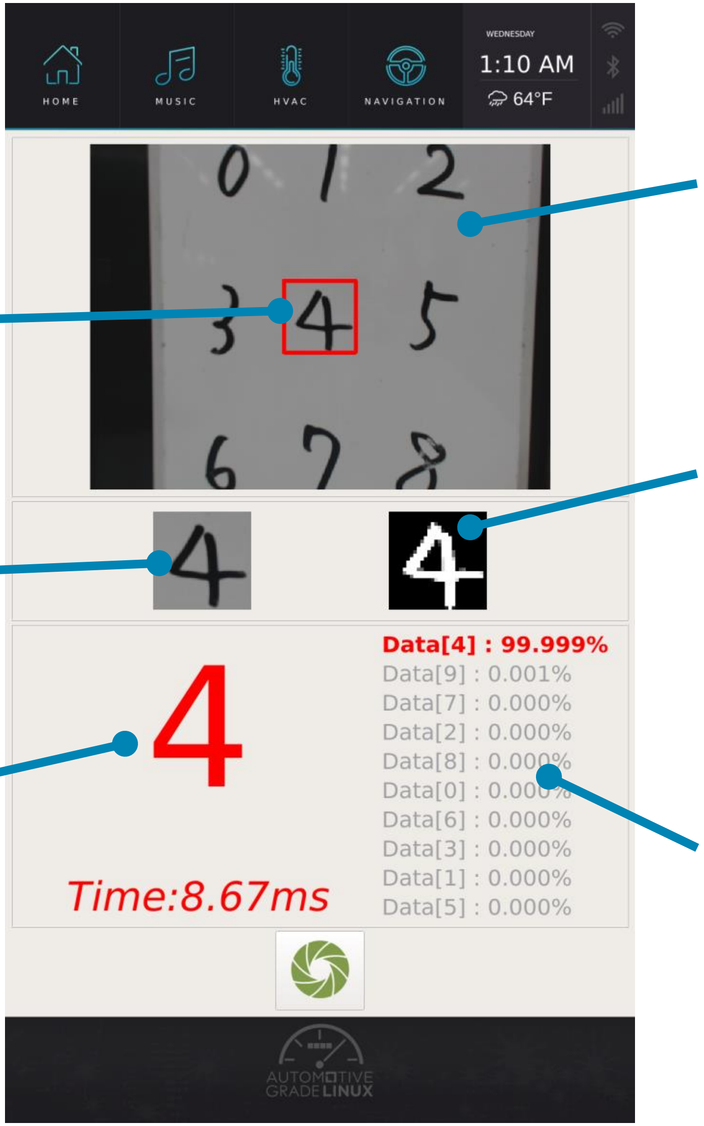
NTT DATA MSE Corporation

Application GUI

Recognized Area
(100 x 100 pixels)

Cropped image
(100 x 100 pixels)

Recognized digit



4

Time:8.67ms

```
Data[4] : 99.999%
Data[9] : 0.001%
Data[7] : 0.000%
Data[2] : 0.000%
Data[8] : 0.000%
Data[0] : 0.000%
Data[6] : 0.000%
Data[3] : 0.000%
Data[1] : 0.000%
Data[5] : 0.000%
```

USB Camera image

Input image to AI Engine
(28 x 28 pixels)

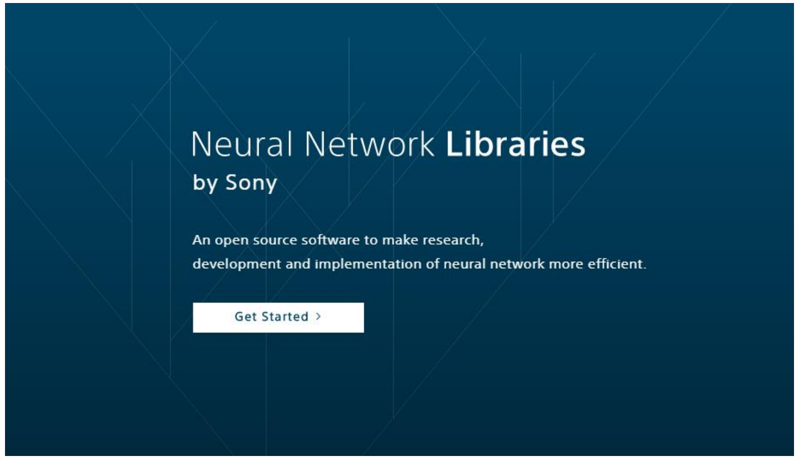
Prediction result from AI Engine
(Probability of each digit)

Neural Network Libraries (NNabla)

Neural Network **Libraries**

https://nnabla.org

- Deep learning framework.
- Intended to be used for research, development and production.
- Aim to have it running everywhere. Deployable to embedded devices.
- Apache License 2.0



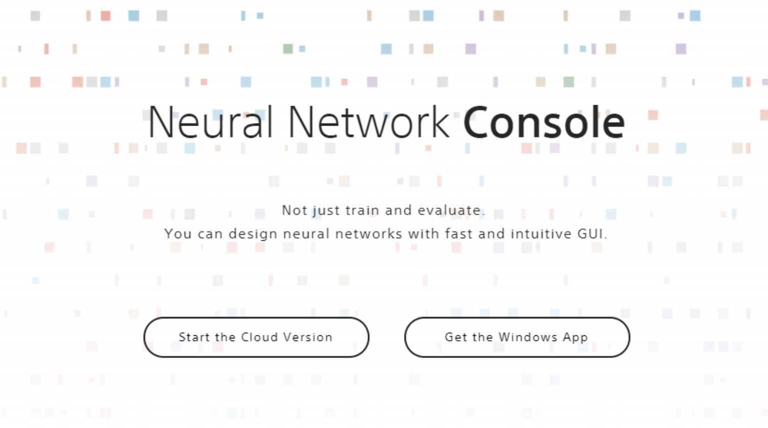
Neural Network **Libraries**
by Sony

An open source software to make research, development and implementation of neural network more efficient.

Neural Network **Console**

https://dl.sony.com

- GUI tool for designing neural networks intuitively.
- Many useful functions to support research and development.
- Trained model can be embedded by using Neural Network Libraries.



Neural Network **Console**

Not just train and evaluate.
You can design neural networks with fast and intuitive GUI.

NTT DaTa
NTT DATA MSE Corporation

Details of the Implementation

Step1: Deep Learning on PC

Step2: Building / Installation

Only for DD **Step3:** Enabling USB webcam

Only for DD **Step4:** Installation of the OpenCV

Step5: App Implementation

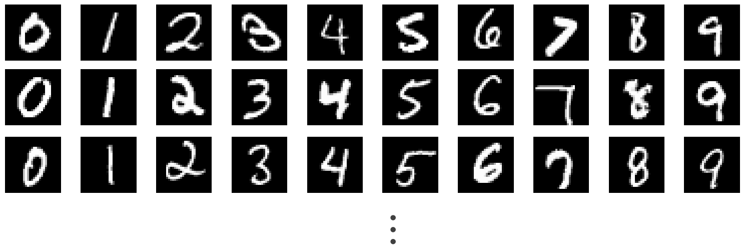
Step3 and 4 are not needed if AGL version is 5.0(EE)

NTT DATA MSE Corporation

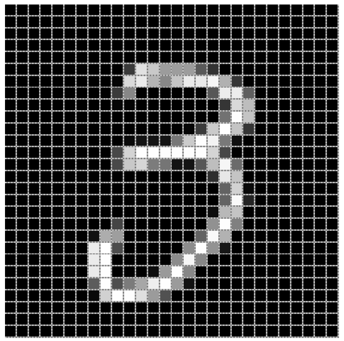
Step1: Deep Learning on PC

Used one of the examples available at the github
→ mnist-collection/classification.py

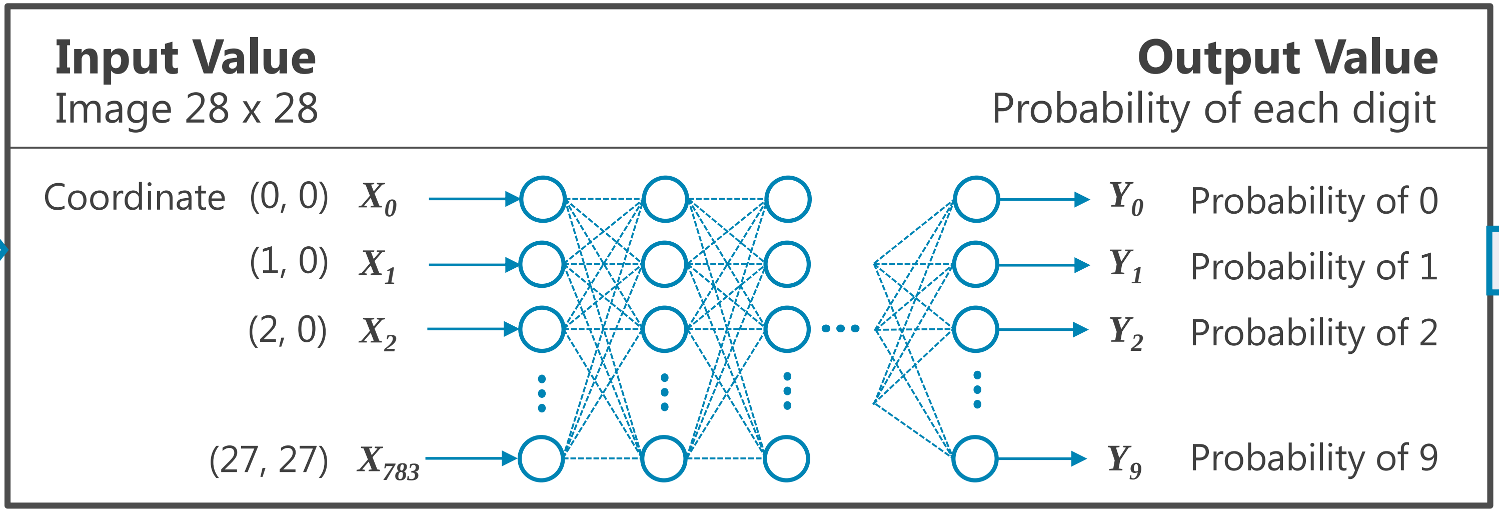
- Downloads and uses MNIST Dataset as training data (60,000 samples of handwritten digits image and label)



- Uses Convolutional Neural Network
  - Input: 28 x 28 pixels grayscale image
  - Output: Prediction of 10-way classification



Input
(28x28)



Input Value
Image 28 x 28

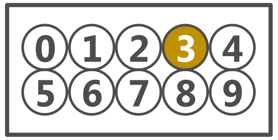
Output Value
Probability of each digit

Coordinate (0, 0) X_0 → Y_0 Probability of 0

(1, 0) X_1 → Y_1 Probability of 1

(2, 0) X_2 → Y_2 Probability of 2

(27, 27) X_{783} → Y_9 Probability of 9



Digit Probability
(10)

NTT DATA MSE Corporation

Step1: Deep Learning on PC

Used one of the examples available at the github → mnist-collection/classification.py

Install the NNabla on a PC

```
$ pip install nnabla
```

Obtain the “mnist-collection”

```
$ cd ~/work/sony/
$ git clone https://github.com/sony/nnabla-examples
```

Obtain examples

Start training

```
$ cd nnabla-examples/mnist-collection
$ python classification.py
2018-01-30 19:42:37,932 [nnabla][INFO]: Initializing CPU extension...
.....
2018-01-30 19:42:38,437 [nnabla][INFO]: Using DataIterator
2018-01-30 19:42:39,343 [nnabla][INFO]: iter=9 {Training loss}=2.30425691605
2018-01-30 19:42:39,343 [nnabla][INFO]: iter=9 {Training error}=0.8375
.....
2018-01-30 19:53:18,056 [nnabla][INFO]: Parameter save (.h5): tmp.monitor/lenet_params_010000.h5
```

Start

Complete

Step2: Building / Installation

Build NNabla for R-Car H3(ARMv8(64bit))

Install cross SDK on a PC (AGL R-Car ARMv8 toolchain)

```
$ wget https://download.automotivelinux.org/AGL/release/eel/5.0.0/m3ulcb-nogfx/deploy/sdk/poky-agl-glibc-x86_64-agl-image-ivi-crosssdk-aarch64-toolchain-5.0.0.sh
$ chmod a+x poky-agl-glibc-x86_64-agl-image-ivi-crosssdk-aarch64-toolchain-5.0.0.sh
$ ./poky-agl-glibc-x86_64-agl-image-ivi-crosssdk-aarch64-toolchain-5.0.0.sh
```

See also:
http://docs.automotivelinux.org/docs/getting_started/en/dev/reference/source-code.html

Build NNabla for R-Car

```
$ git clone https://github.com/sony/nnabla
$ source /opt/poky-agl/5.0.0/environment-setup-aarch64-agl-linux
$ mkdir -p nnabla/build && cd nnabla/build
$ cmake .. -DBUILD_CPP_UTILS=ON -DBUILD_PYTHON_PACKAGE=OFF
$ make
$ ls -l lib/
-rwxrwxr-x 1 nttdmse nttdmse 191170296 12月 20 06:21 libnnabla.so
-rwxrwxr-x 1 nttdmse nttdmse  25392344 12月 20 06:21 libnnabla_utils.so
```

Obtain source code

Setup environment

Build

See also:
https://nnabla.readthedocs.io/en/latest/cpp/installation.html

NTT DATA MSE Corporation

Step2: Building / Installation

Install the built shared libraries of the NNabla and the pre-trained model to the target filesystem.

Shared libraries of the NNabla

```
$ export SDCARD=/tmp/agl
$ sudo mount /dev/sdc1 $SDCARD
$ sudo cp libnnabla.so $SDCARD/usr/lib/
$ sudo cp libnnabla_utils.so $SDCARD/usr/lib/
```

Copy

Pre-trained model

```
$ cd ~/work/sony/nnabla/examples/cpp/mnist_runtime
$ NNABLA_EXAMPLES_ROOT=~/work/sony/nnabla-examples python save_nnp_classification.py
$ ls -l lenet_010000.nnp
-rw-rw-r-- 1 nttdmse nttdmse 86920  1月 29 19:16 lenet_010000.nnp

$ sudo cp lenet_010000.nnp $SDCARD/home/data/
$ sync
$ sudo umount $SDCARD
```

Convert to NNabla file format (NNP)

Copy

Step3: Enabling USB webcam

Only for DD

Enable the USB Video Class in order to use USB webcam on R-Car H3.

Enable the UVC (USB Video class) of kernel config

```
$ export MACHINE=h3ulcb
$ source meta-agl/scripts/aglsetup.sh -m $MACHINE -b build-renesas-kernel agl-devel agl-demo agl-netboot agl-appfw-smack
agl-localdev

$ bitbake virtual/kernel

$ bitbake linux-renesas –c menuconfig

Device Drivers --> Multimedia support  ---> [*] Media USB Adapters
                [*] USB Video Class (UVC)
                [*] UVC input events device support (NEW)

$ bitbake virtual/kernel
```

Setup environment

Build Kernel

Change kernel configuration

Rebuild kernel

Update kernel image

```
$ sudo cp tmp/deploy/images/m3ulcb/Image--4.9.0+git0+098ccf1c9b-r1-m3ulcb-20171116044641.bin SDCARD/boot/Image-4.9.0-
yocto-standard
```

Step4: Installation of the OpenCV

Only for DD

Install the OpenCV to handle camera images

- OpenCV = Open Source Computer Vision Library
- Used for obtaining image from the webcam and pre-process the acquired images before passing them to the NNabla.

Build the OpenCV

```
$ export MACHINE=h3ulcb
$ source meta-agl/scripts/aglsetup.sh -m $MACHINE -b build-opencv agl-devel agl-demo agl-netboot agl-appfw-smack agl-localde

$ bitbake opencv
```

Setup environment

Build

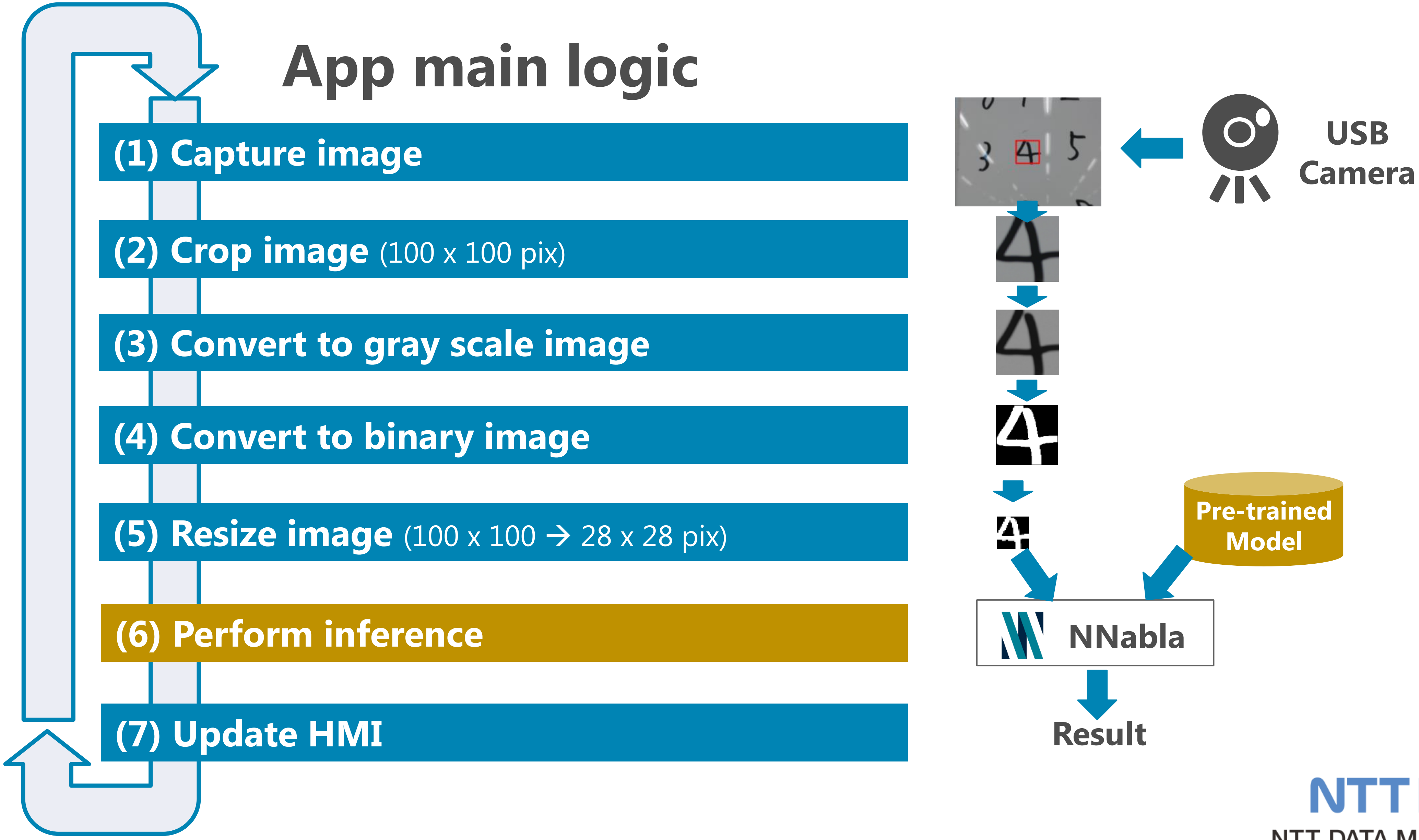
Install built binaries to target filesystem

```
$ sudo cp -a ./tmp/work/aarch64-agl-linux/opencv/3.2+gitAUTOINC+70bbf17b13-r0/image/* $SDCARD/
$ sudo cp -a ./tmp/work/aarch64-agl-linux/v4l-utils/1.12.3-r0/image/* $SDCARD/
$ sudo cp -a ./tmp/work/aarch64-agl-linux/libwebp/0.6.0-r0/image/* $SDCARD/

$ sync; sudo umount $SDCARD
```

Copy binaries

Step5: App Implementation

App main logic



(1) **Capture image**

(2) **Crop image** (100 x 100 pix)

(3) **Convert to gray scale image**

(4) **Convert to binary image**

(5) **Resize image** (100 x 100 → 28 x 28 pix)

(6) **Perform inference**

(7) **Update HMI**

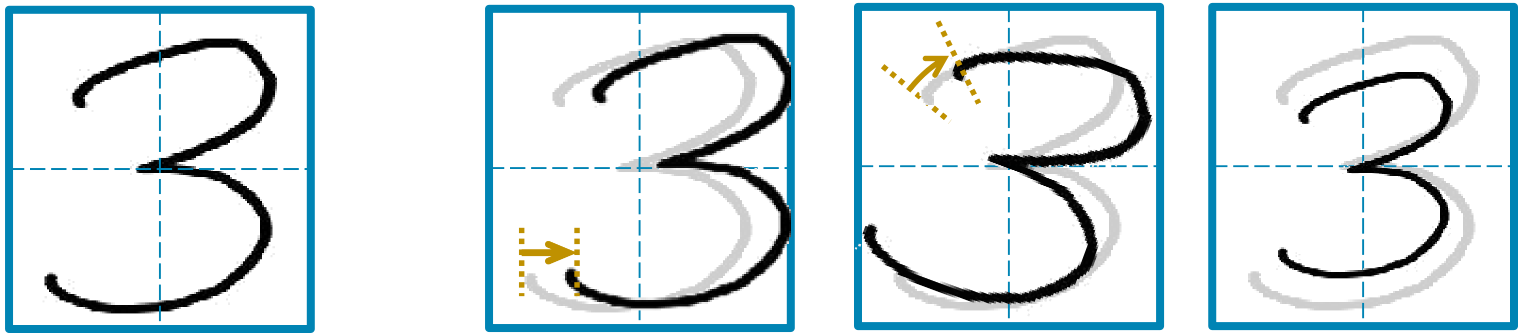
Step5: App Implementation(Source Code)

```
// (0) Open Device
cv::VideoCapture cap("/dev/video22", cv::CAP_V4L2);    // videoN: Set according to the environment. This is R-Car H3 with AGL5.0.0.

// (1) Capture image
cv::Mat frame;
cap >> frame;

// Pre-process image
// (2) Crop video image [100x100pix]
cv::Rect rect(GET_VIEW_SIZE_LEFT, GET_VIEW_SIZE_TOP, GET_VIEW_SIZE_WIDTH, GET_VIEW_SIZE_HEIGHT);
cv::Mat rectImg(frame, rect);

// (3) Convert to gray scale image
cv::Mat grayImg;
cv::cvtColor(rectImg, grayImg, CV_RGB2GRAY);

// (4) Convert to binary image ( Invert | Threshold = 127)
cv::Mat binImg;
cv::threshold(grayImg, binImg, 127, 255, cv::THRESH_BINARY_INV);

// (5) Resize image [100x100pix -> 28x28pix]
cv::Mat resizeImg;
cv::resize(binImg, resizeImg, cv::Size(), PGM_WIDTH/grayImg.cols ,PGM_HEIGHT/grayImg.rows);

// Add pgm header
pgmformat((char *)"cap.pgm",resizeImg);

// (6) Perform inference
int prediction = 0;
float score[10] = {};
double elapsed = 0;
const std::string nnp_filepath[] = "/home/data/lenet_010000.nnp";
const std::string pgmImageName[] = "cap.pgm";
prediction = EdgeAiNNblaMnistRuntime(pgmImageName, nnp_filepath, score, &elapsed);    // nnabla wrapper API
```

Experiment Result

Recognition Result (First Try)

Not possible to recognize digits for the following patterns



Recognized

Not Recognized

Shifted

Rotated

Scaled

Recognition Result (After improvement)

■ Used Data Augmentation to improve pre-trained model

- Randomly alters the MNIST digit images when perform training.
  → scaling, rotation, aspect ratio, distorting, brightness, contrast, add noise
- Updated the pre-trained model on the device.
- Recognition rate has been improved.
  - ➢ Tested with 1,000 cases. (Handwritten digits by our colleague)
  - ➢ Counted as "Recognized" if the probability is larger than 50%

Before

86.0%

(860/1,000)

→

After

94.8%

(948/1,000)

■ Improvement of predictions are possible by updating pre-trained models

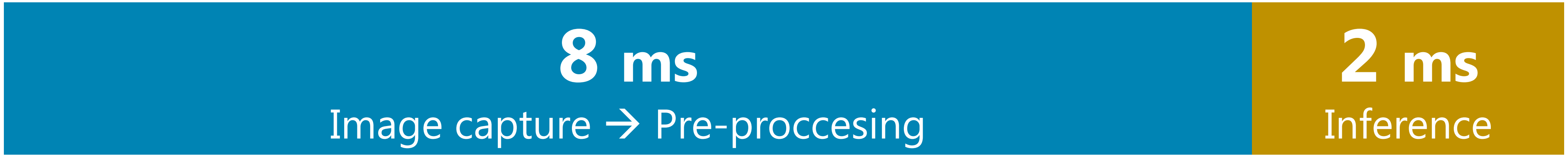
- Delivering up-to-date pre-trained models to devices can provide more accurate prediction results for users.

NTT DATA

NTT DATA MSE Corporation

Performance

Processing Time



8 ms

Image capture → Pre-proccesing

2 ms

Inference

*Measured on R-Car H3

Resource Usage

#	Item	Value
1	CPU usage rate	55 %
2	Memory [RSS] usage	80 MB
	Size of the Pre-trained Model	85 KB

*Measured on R-Car H3

Conclusion

Summary

- **Explained how we implemented the Handwritten Digit Recognition App**
  - Performed deep learning on a PC
  - Deployed pre-trained model to R-Car H3 board
  - Variation of training data set affects inference results
  - Updating of pre-trained model can improve inference results

- **It wasn’t difficult than expected to take the first step toward using AI engine on AGL**
  - Implemented the app in 3 weeks
  - Please try it!

Summary

Next Step

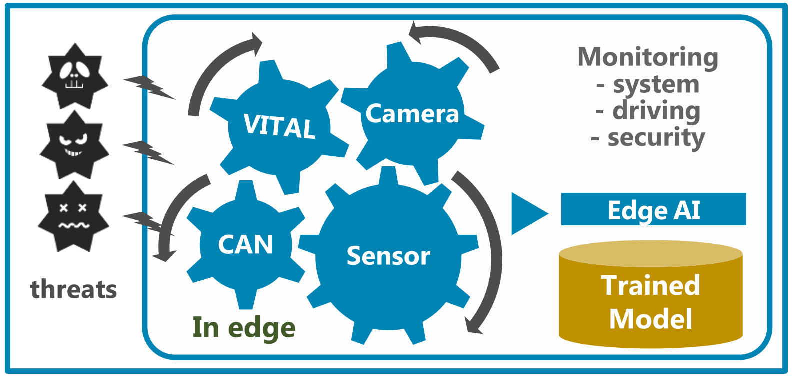
1. Use other types of data for deep learning

- Create own neural network architecture
- Use information other than image available in vehicle

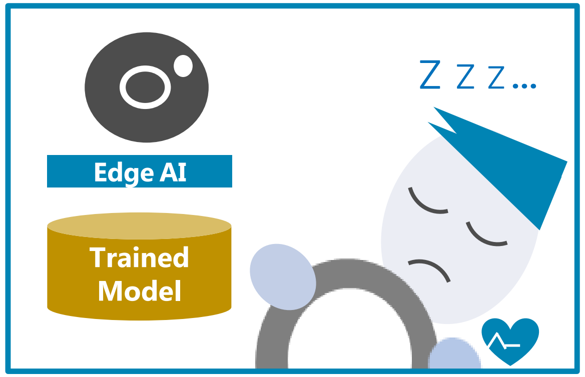
2. Performance improvement

- More complicated neural network architecture causes performance issues
- Utilize GPU to accelerate calculations

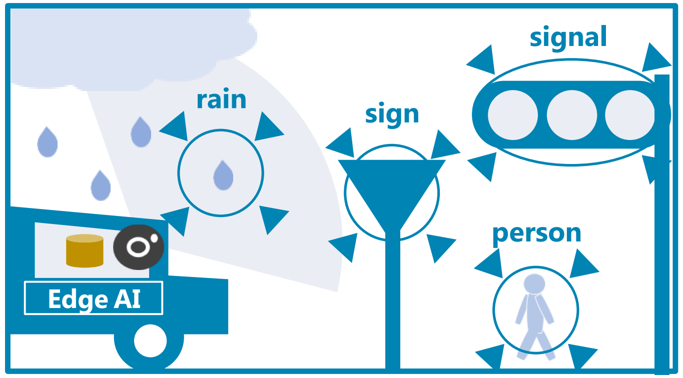
Future use-cases



Use various information of embedded devices

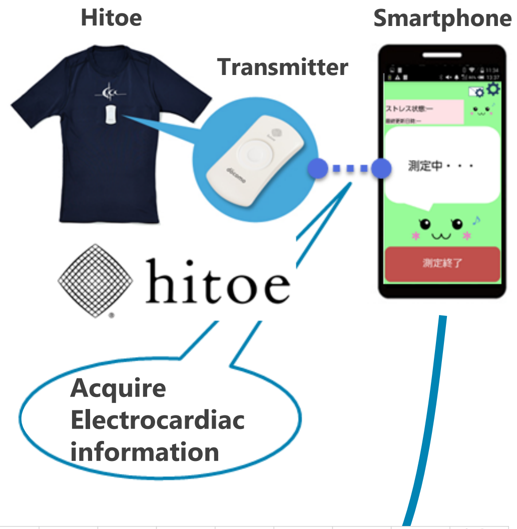


Detect driver's drowsiness

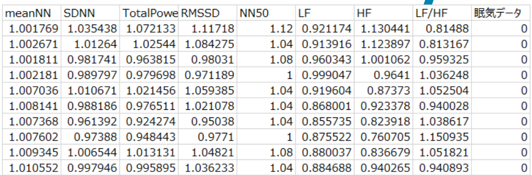


Improvement of autonomous driving technology

Drowsiness Detection

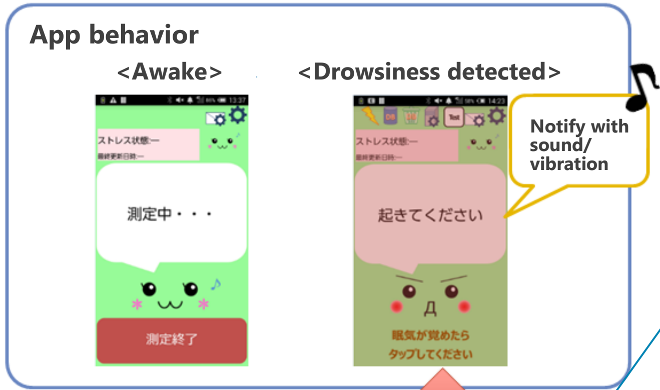


Acquire Electrocardiac information

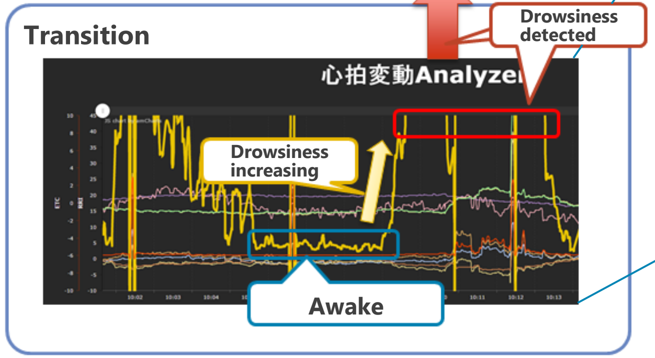


meanNN	SDNN	TotalPowe	RMSSD	NN50	LF	HF	LF/HF	眠気データ
1.001769	1.035438	1.072133	1.11718	1.12	0.921174	1.130441	0.81488	0
1.002671	1.01264	1.02544	1.084275	1.04	0.913916	1.123897	0.813167	0
1.001811	0.981741	0.963815	0.98031	1.08	0.960343	1.001062	0.959325	0
1.002181	0.989797	0.979698	0.971189	1	0.999047	0.9641	1.036248	0
1.007036	1.010671	1.021456	1.059385	1.04	0.919604	0.87373	1.052504	0
1.008141	0.988186	0.976511	1.021078	1.04	0.868001	0.923378	0.940028	0
1.007368	0.961392	0.924274	0.95038	1.04	0.855735	0.823918	1.038617	0
1.007602	0.97388	0.948443	0.9771	1	0.875522	0.760705	1.150935	0
1.009345	1.006544	1.013131	1.04821	1.08	0.880037	0.836679	1.051821	0
1.010552	0.997946	0.995895	1.036233	1.04	0.884688	0.940265	0.940893	0

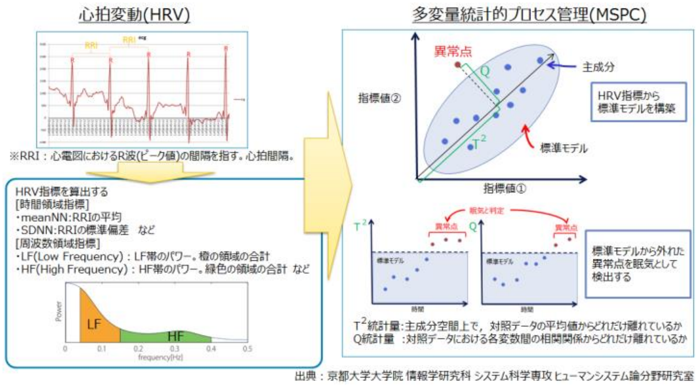
App behavior



Transition



Drowsiness Detection Algorithm Based on Heart Rate Variability



※RRI：心電図におけるR波(ピーク値)の間隔を指す。心拍間隔。

HRV指標を算出する
[時間領域指標]
・meanNN:RRIの平均
・SDNN:RRIの標準偏差　など
[周波数領域指標]
・LF(Low Frequency)：LF帯のパワー。橙の領域の合計
・HF(High Frequency)：HF帯のパワー。緑色の領域の合計　など

HRV指標から
標準モデルを構築

標準モデルから外れた
異常点を眠気として
検出する

T^2統計量:主成分空間上で，対照データの平均値からどれだけ離れているか
Q統計量　:対照データにおける各変数間の相関関係からどれだけ離れているか

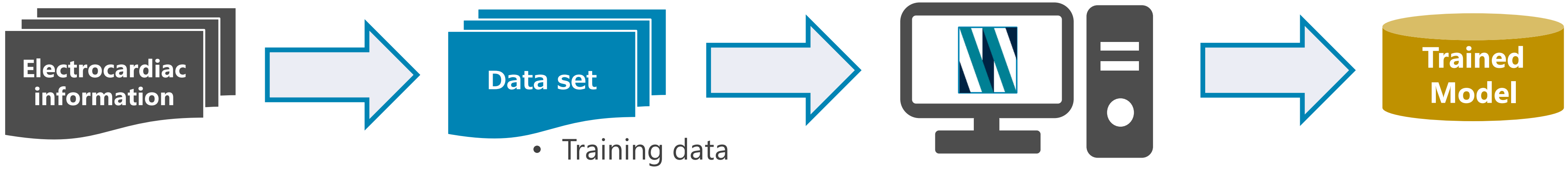
出典：京都大学大学院 情報学研究科 システム科学専攻 ヒューマンシステム論分野研究室

Trial on-going:
Replace the algorithm with AI

Prepare vital data

Create data set

Training



Electrocardiac information

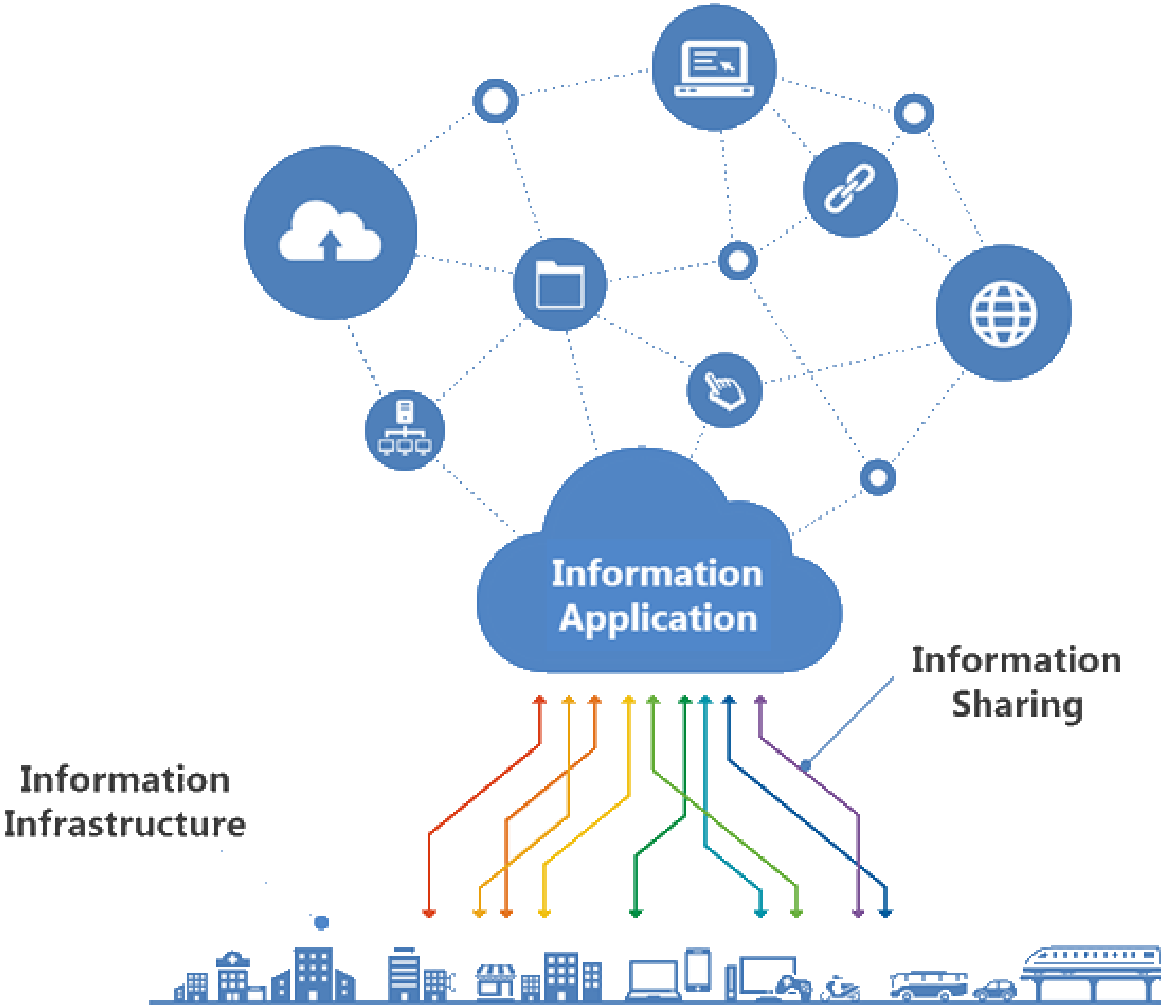
Data set

- Training data
- Validation data

Trained Model

Questions?

Thank you very much!!



Smart Life Community®

NTT DATA MSE Corporation

NTT DATA

Global IT Innovator