Practical Verification for Edge AI use and Effort for Functional Improvement

June 20st, 2018

Yasumitsu Takahashi
NTT DATA MSE Corporation

NTT DATA
NTT DATA MSE Corporation

Who am I?

NTT DATA MSE Corporation

Name	**Yasumitsu Takahashi**
Position	**Deputy Manager**
Carrier	**- Technical leader of R&D projects in NTT DATA MSE** **- Linux-based embedded devices** **- Application, middleware, kernel development**

NTT DATA

NTT DATA MSE Corporation

Agenda

- **Introduction to our Activities**
- **Effort for performance improvement**
  - ✓ System Overview
  - ✓ How to integrate app utilizing OpenCL into AGL
  - ✓ Performance Verification & Consideration
- **Possibility verification of continuous improvement of inference accuracy for edge AI**
  - ✓ System Overview
  - ✓ Demonstration & Verification Result

Introduction to our Activities

Our team have started AI related activities since October 2017

- There are AI related news almost everyday
- First Step: Let's use an AI engine on edge devices
  - ✓ Implemented a demo system "Handwritten Digit Recognition App"
- Try creating own neural network architecture
- Performance improvement
  - ✓ More complicated neural network architecture and graphic processing cause performance issues
- Consider how to improve inference accuracy on a practical scene and update calibration such as personal preference

In this presentation

- Performance verification
  - ✓ How to integrate app utilizing OpenCL(GPU)
  - ✓ Verify the performance effect of OpenCL
- Verification to improve inference accuracy for edge AI continuously

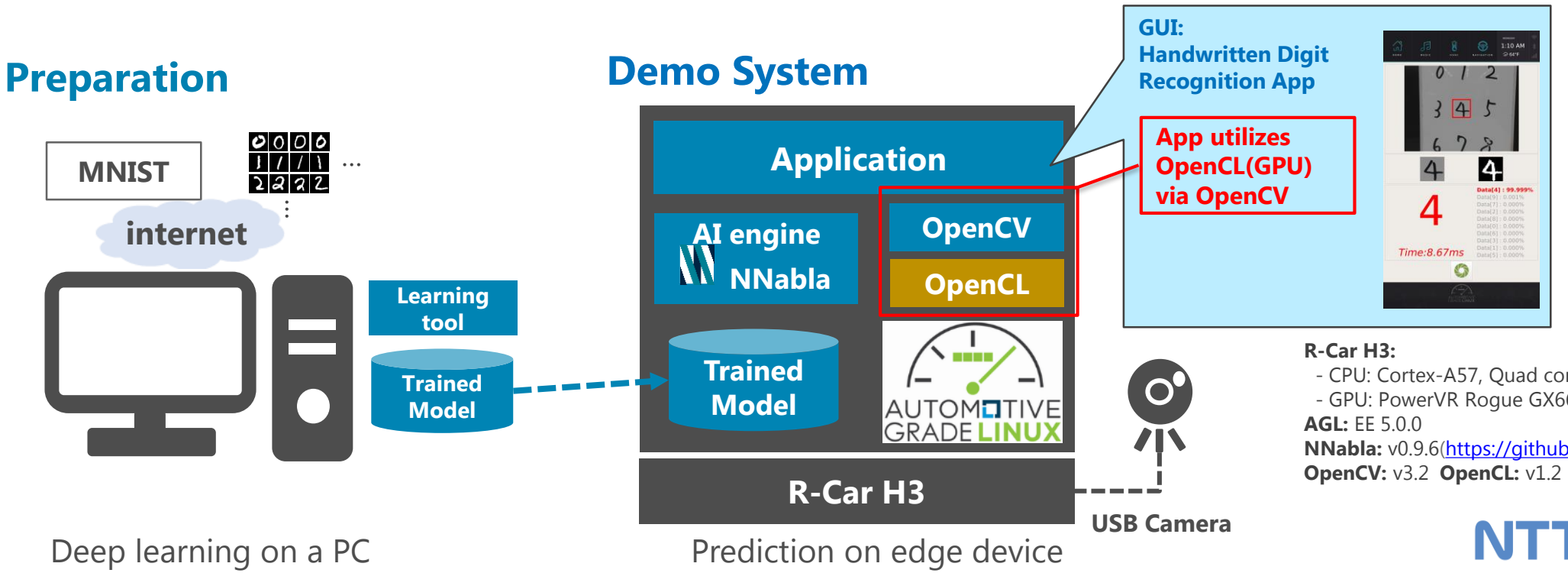
NTT DATA MSE Corporation

Effort for performance improvement

System Overview (1/2)

Perform the followings to the pre-developed demo system (Handwritten Digit Recognition)

- Investigate how to apply OpenCL (GPU) and implement it
- Verify the effect of OpenCL



R-Car H3:
- CPU: Cortex-A57, Quad core, 1.500 GHz
- GPU: PowerVR Rogue GX6650, 192 core, 600MHz

AGL: EE 5.0.0
NNabla: v0.9.6(https://github.com/sony/nnabla)
OpenCV: v3.2 **OpenCL:** v1.2

NTT DATA MSE Corporation

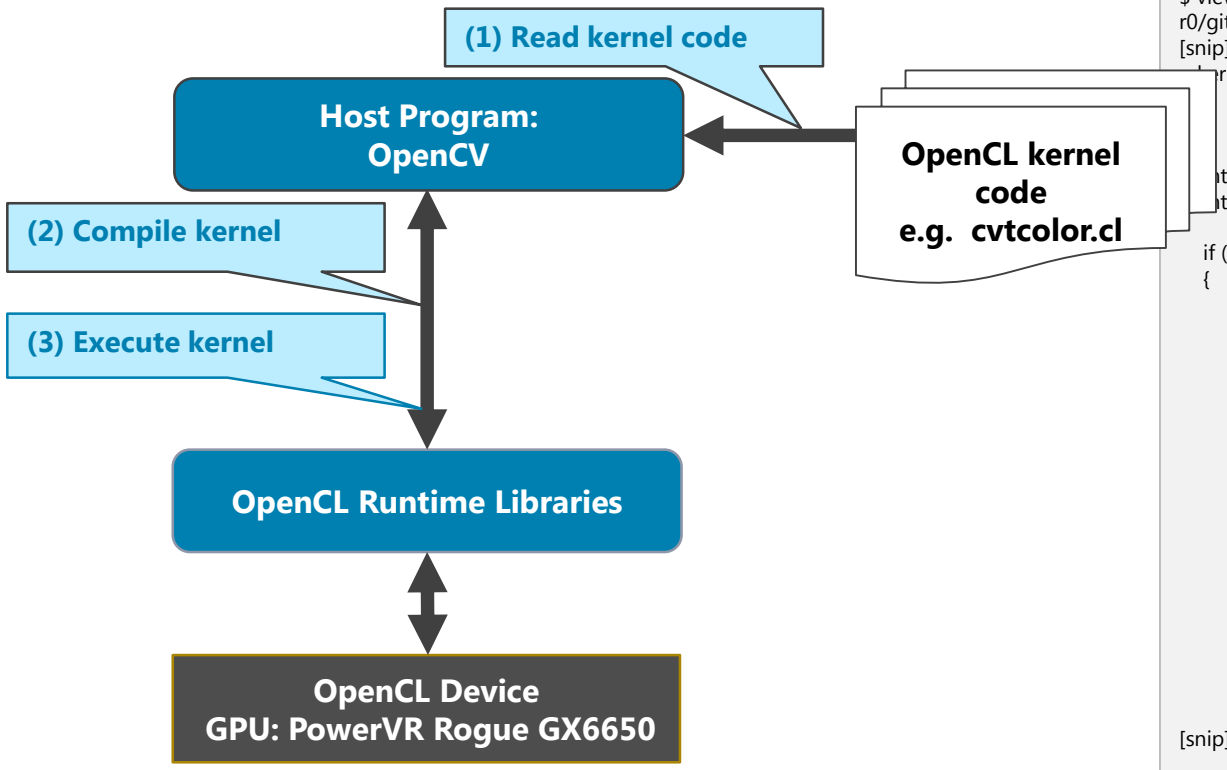
System Overview (2/2)

What is OpenCL?

OpenCL（=Open Computing Language) is a cross-platform framework for parallel computing by using computing resources mixed by multicore CPU/GPU/DPS (Heterogeneous environment) in Open CL C-language

Look easy to learn on C-language basis.
Look possible to secure the portability of source code by not depending to the specific HW PF.

OpenCL Compilation Overview




```
$ view build-h3-cl/tmp/work/aarch64-agl-linux/opencv/3.2+gitAUTOINC+70bbf17b13-
r0/git/modules/imgproc/src/opencl/cvtcolor.cl
[snip]
rnel void RGB2YUV(__global const uchar* srcptr, int src_step, int src_offset,
                  __global uchar* dstptr, int dst_step, int dt_offset,
                  int rows, int cols)

t x = get_global_id(0);
t y = get_global_id(1) * PIX_PER_WI_Y;

  if (x < cols)
  {
    int src_index = mad24(y, src_step, mad24(x, scnbytes, src_offset));
    int dst_index = mad24(y, dst_step, mad24(x, dcnbytes, dt_offset));

    #pragma unroll
    for (int cy = 0; cy < PIX_PER_WI_Y; ++cy)
    {
      if (y < rows)
      {
        __global const DATA_TYPE* src = (__global const DATA_TYPE*)(srcptr + src_index);
        __global DATA_TYPE* dst = (__global DATA_TYPE*)(dstptr + dst_index);
        DATA_TYPE_4 src_pix = vload4(0, src);
        DATA_TYPE b = src_pix.B_COMP, g = src_pix.G_COMP, r = src_pix.R_COMP;

        __constant float * coeffs = c_RGB2YUVCoeffs_f;
        const DATA_TYPE Y = fma(b, coeffs[0], fma(g, coeffs[1], r * coeffs[2]));
        const DATA_TYPE U = fma(b - Y, coeffs[3], HALF_MAX_NUM);
        const DATA_TYPE V = fma(r - Y, coeffs[4], HALF_MAX_NUM);
[snip]
```

How to integrate app utilizing OpenCL into AGL (1/3)

Prepare OpenCL runtime libraries(proprietaries) supporting R-Car H3 provided by Renesas

Prepare OpenCL runtime libraries

(1) Execute 1. to 10. by following the steps in R-Car/Boards/Yocto-Gen3(https://elinux.org/R-Car/Boards/Yocto-Gen3) (Details omitted)

(2) Execute the steps up to "Building Yocto images" by following the ones in R-Car/Boards/Yocto-Gen3(https://elinux.org/R-Car/Boards/Yocto-Gen3/OpenCL) (Details omitted)

(3) Install OpenCL runtime to AGL target filesystem(SDCARD)

```
$ sudo cp -a ./build/tmp/sysroots/h3ulcb/usr/lib/libOpenCL.so $SDCARD/usr/lib/
$ sudo cp -a ./build/tmp/sysroots/h3ulcb/usr/lib/libPVROCL.so $SDCARD/usr/lib/
$ sudo cp -a ./build/tmp/work/h3ulcb-poky-linux/cl-gles-user-module/1.0-r0/image/usr/local/bin/ocl_unit_test $SDCARD/usr/bin/
$ sudo cp -a ./build/tmp/sysroots/h3ulcb/usr/lib/liboclcompiler.so $SDCARD/usr/lib/
$ sudo cp -a ./build/tmp/sysroots/h3ulcb/usr/lib/libglslcompiler.so $SDCARD/usr/lib/
$ sudo cp -a ./build/tmp/sysroots/h3ulcb/usr/lib/libdbm.so $SDCARD/usr/lib/
$ sudo cp -a ./build/tmp/sysroots/h3ulcb/usr/lib/lib/libdlc_REL.so $SDCARD/usr/lib/
$ sudo cp -a ./build/tmp/sysroots/h3ulcb/usr/lib/lib/libufwriter.so $SDCARD/usr/lib/
```

ocl_unit_test can check the build of OpenCL(GPU) runtime environment. For example, if any libraries like CL compiler (liboclcompiler.so) are short, the test will result in Fail…

About AGL Getting Started. See also:
http://docs.automotivelinux.org/docs/getting_started/en/dev/reference/source-code.html
http://docs.automotivelinux.org/docs/getting_started/en/dev/reference/machines/R-Car-Starter-Kit-gen3.html

NTT DATA MSE Corporation

How to integrate app utilizing OpenCL into AGL (2/3)

Need to enable OpenCL option of OpenCV and rebuild

Build OpenCV with OpenCL enabled

(1) Clean-up opencv & configure

In case of pre-built, clean once and execute up to config.

```
$ bitbake opencv -c clean
$ bitbake opencv -c configure
```

(2) Enable OPENCL option

Enable OPENCL option in Cmake environment variable definition file (otherwise, the error occurs in bitbake)

```
$ vi tmp/work/aarch64-agl-linux/opencv/3.2+gitAUTOINC+70bbf17b13-r0/build/CMakeVars.txt
WITH_OPENCL=OFF  →  WITH_OPENCL=ON

$ vi tmp/work/aarch64-agl-linux/opencv/3.2+gitAUTOINC+70bbf17b13-r0/build/CMakeCache.txt
WITH_OPENCL:BOOL=ON  →  WITH_OPENCL:BOOL=ON
```

(3) Modify cvconfig.h

Define HAVE_OPENCL in config header file (probably, either of the followings is ok)

```
$ vi tmp/work/aarch64-agl-linux/opencv/3.2+gitAUTOINC+70bbf17b13-r0/build/cvconfig.h
#define HAVE_OPENCL
$ vi tmp/work/aarch64-agl-linux/opencv/3.2+gitAUTOINC+70bbf17b13-r0/build/opencv2/cvconfig.h
#define HAVE_OPENCL
```

(4) Rebuild opencv & install

```
$ bitbake opencv -c compile
$ bitbake opencv -c install
$ sudo cp -a build-h3-cl/tmp/work/aarch64-agl-linux/opencv/3.2+gitAUTOINC+70bbf17b13-r0/image/* $SDCARD/
```

How to integrate app utilizing OpenCL into AGL (3/3)

Modify App code in order to use OpenCL(GPU) from OpenCV

Walk through modified demo application code

```
// Video(camera) capture
   .....
   // Pre-processing image[(1) - (5)] for AI engine
   // (1) Cut out video image into a rectangle [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);

   // (2) Convert gray scale
   //cv::Mat grayImg;
   //cv::cvtColor(rectImg, grayImg, CV_RGB2GRAY);
    cv::UMat u_rectImg, u_grayImg;
    rectImg.copyTo(u_rectImg);
    cv::cvtColor(u_rectImg, u_grayImg, CV_RGB2GRAY);

   // (3) Convert binary image
   //cv::Mat binImg;
   //cv::threshold(grayImg, binImg, 127, 255, cv::THRESH_BINARY_INV);
    cv::UMat u_binImg;
    cv::threshold(u_grayImg, u_binImg, 127, 255, cv::THRESH_BINARY_INV);

   // (4) Resize binary image[100x100pix -> 28x28pix]
   //cv::Mat resizeImg;
   //cv::resize(binImg, resizeImg, cv::Size(), PGM_WIDTH/grayImg.cols ,PGM_HEIGHT/grayImg.rows);
   cv::UMat u_resizeImg;
   cv::resize(u_binImg, u_resizeImg, cv::Size(), PGM_WIDTH/u_grayImg.cols ,PGM_HEIGHT/u_grayImg.rows);

   // (5) Add pgm header
   .....
   // AI engine inference with MNIST classification model
    .....
```

Just replace Mat class (matrix data and data property) with UMat class!
When OpenCL(GPGPU) is available in Target environment, switch to GPU operation, otherwise, to CPU operation.
Alternatively, it is able to switch to CPU/GPU by "cv::ocl::setUseOpenCL(false/true)".

Performance Verification (1/5)

Measured demo application

GPU/CPU	Processing time (a)	CPU load (b)	Remarks
GPU	11 ms	57.8 %	a : UI visual check for a set of processing time in AI demo b : Average on "CPUs utilized“ measured 3 times by ”perf stat -p PID[AI demo] -- sleep 10”
CPU	9 ms	57.2 %	

Faster process for CPU use. No difference for CPU load.

Breakdown of measured (Function utilized OpenCL)

OpenCL(GPU) supporting process	GPU/CPU	Processing time	CPU load	Remarks
Grayscale conversion[100x100pix]	GPU	1.325 ms	N/A	**8 times slower for GPU use**
	CPU	0.168 ms	N/A	
Binary conversion[100x100pix]	GPU	0.440 ms	N/A	**16 times slower for GPU use**
	CPU	0.028 ms	N/A	
Resize [100x100pix -> 28x28pix]	GPU	0.364 ms	N/A	**9 times slower for GPU use**
	CPU	0.042 ms	N/A	

OpenCL(GPU) effect cannot be recognized for image process used in pre-developed demo. Approx 10 times slower for each supporting process.

Assumed “GPU effect cannot be worked due to the small image size (operating volume)” in pre-developed demo case. Re-verify each OpenCL supporting process by enlarging the image size (VGA/Full-HD).

Performance Verification (2/5)

Verify by measuring the processing time[ms] & CPU load[%] of 100 times target process calls. Average value on 3 times turns out result.

Example test code: cvtColor(BGR2GRAY)

```
int main(int argc, char *argv[])
{
[snip]
    if( false == parser.get<bool>("openCL") ) {  // Not use OpenCL[use CPU] if command line param is false.
        cv::ocl::setUseOpenCL(false);
        clMode = "CPU";
    }
    cv::Mat frame = cv::imread(parser.get<string>("input"), cv::IMREAD_COLOR);  // input image data
    const int countN = parser.get<int>("countN");
    double f = 1000.0f / cv::getTickFrequency();

    // for UMat
    cout<<“Convert gray scale."<<endl;
    int64 start = cv::getTickCount();
    cv::UMat u_grayImg, u_frame;
    frame.copyTo(u_frame);
    for(int i =0; i < countN; i++) {
        cv::cvtColor(u_frame, u_grayImg, CV_RGB2GRAY);  // Verification target
    }
    int64 end = cv::getTickCount();
    cout << "[UMat:"<<clMode<<"] " << (end - start) * f << "[ms]" << endl;
    return 0;
}
```

Build & Install to target filesystem

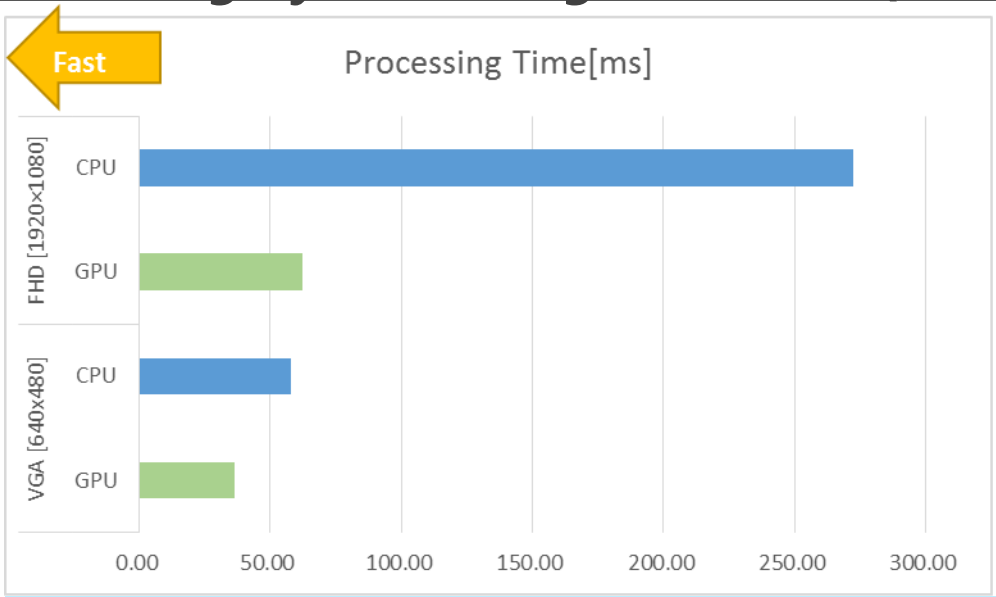
```
$ source /opt/poky-agl/5.0.0/environment-setup-aarch64-agl-linux
$ $CXX TestOpenCL_cvtColor.cpp -I$SDKTARGETSYSROOT/usr/include/opencv2 -L$SDKTARGETSYSROOT/usr/lib -lopencv_core -lopencv_imgcodecs -lopencv_imgproc -o TestOpenCL_cvtColor
$ sudo cp TestOpenCL_cvtColor $SDCARD/usr/bin/
```

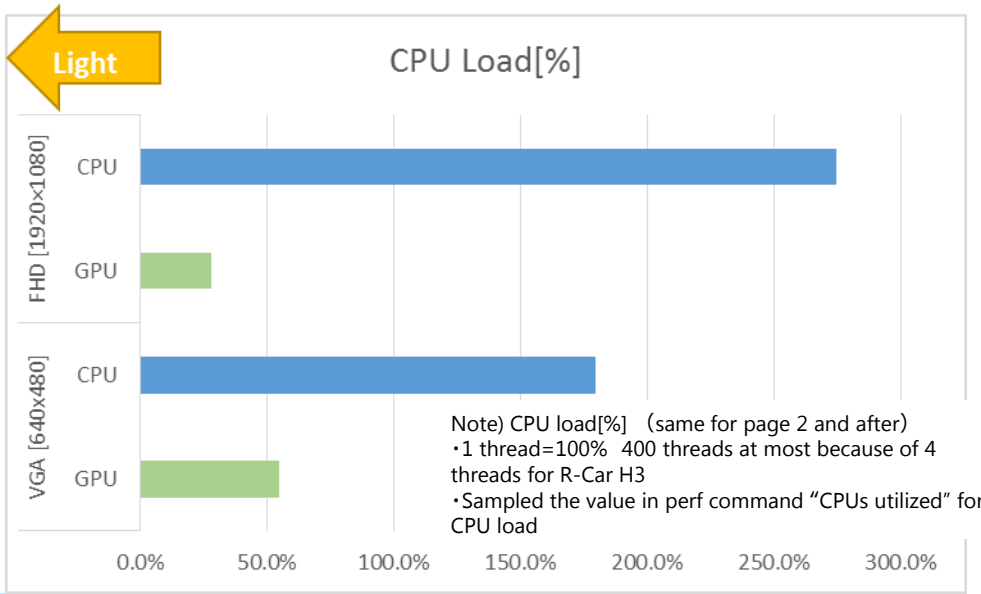
Perform test code and measure performance

```
root@h3ulcb:~# perf stat TestOpenCL_cvtColor --cl=true -i=/home/data/testdata-vga.png -o=/home/data/testdata-vga-out-gpu.png
```

Performance Verification (3/5)

Convert to gray scale image : cvtColor(BGR2GRAY)





Note) CPU load[%] （same for page 2 and after）
・1 thread=100% 400 threads at most because of 4 threads for R-Car H3
・Sampled the value in perf command “CPUs utilized” for CPU load

GPU supporting effect works for both processing time & CPU load after VGA size!
Shortened processing time by ”58ms->36ms” and lowered CPU load by ”178%->55%” in VGA.
GPU effects work more in Full-HD because of shortening processing time by ”272ms->62ms” and lowering CPU load by ”274%->28%”.

Snap shot）Case CPU／ Full-HD

Performance counter stats for 'TestOpenCL_cvtColor --cl=false ,,,

1090.676095	task-clock (msec)	# 2.733 CPUs utilized
1106	context-switches	# 0.001 M/sec
22	cpu-migrations	# 0.020 K/sec
2703	page-faults	# 0.002 M/sec
1631814102	cycles	# 1.496 GHz
3582132759	instructions	# 2.20 insn per cycle
<not supported>	branches	
1844910	branch-misses	# 0.00% of all branches

Snap shot）Case GPU／ Full-HD

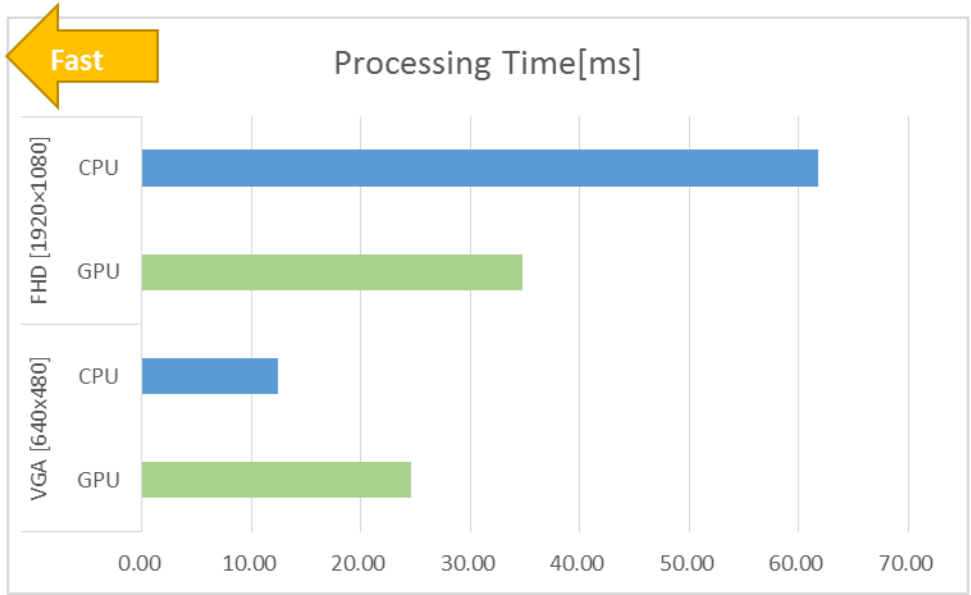
Performance counter stats for 'TestOpenCL_cvtColor --cl=true ,,,

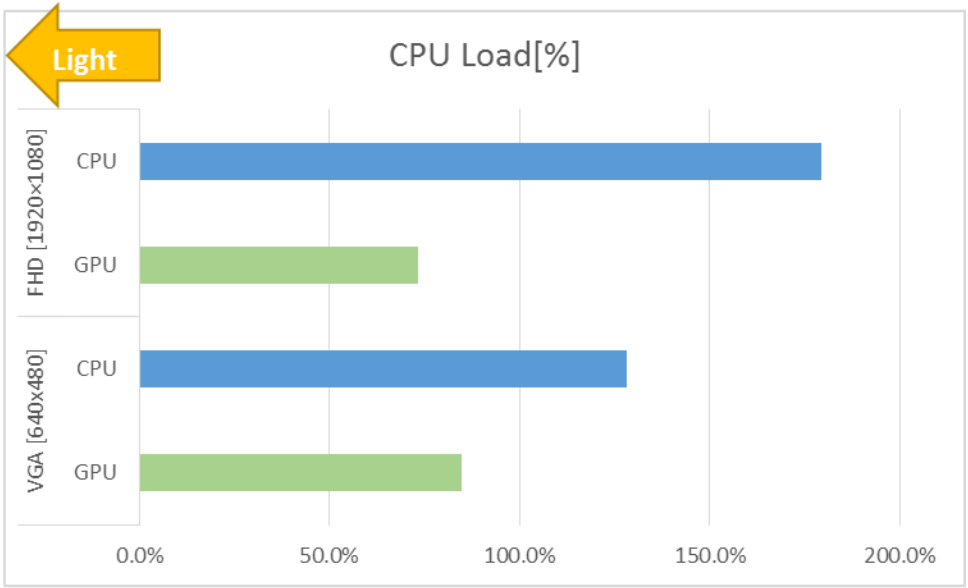
211.759924	task-clock (msec)	# 0.280 CPUs utilized
305	context-switches	# 0.001 M/sec
0	cpu-migrations	# 0.000 K/sec
8609	page-faults	# 0.041 M/sec
316880448	cycles	# 1.496 GHz
306382990	instructions	# 0.97 insn per cycle
<not supported>	branches	
1984822	branch-misses	# 0.00% of all branches

NTT DATA MSE Corporation

Performance Verification (4/5)

Convert to binary image : threshold(THRESH_BINARY_INV)





Longer processing time but lower CPU load by "128%->85%" in VGA.
Shorter processing time by "62ms->35ms" and lower CPU load by "173%->106%" in Full-HD and GPU effects work for both!

Snap shot）Case CPU／ Full-HD

Performance counter stats for 'TestOpenCL_threshold --cl=false ,,,

315.582294	task-clock (msec)	# 1.795 CPUs utilized
1130	context-switches	# 0.004 M/sec
10	cpu-migrations	# 0.032 K/sec
2742	page-faults	# 0.009 M/sec
469894072	cycles	# 1.489 GHz
353487896	instructions	# 0.75 insn per cycle
<not supported>	branches	
1320894	branch-misses	# 0.00% of all branches

Snap shot）Case GPU／ Full-HD

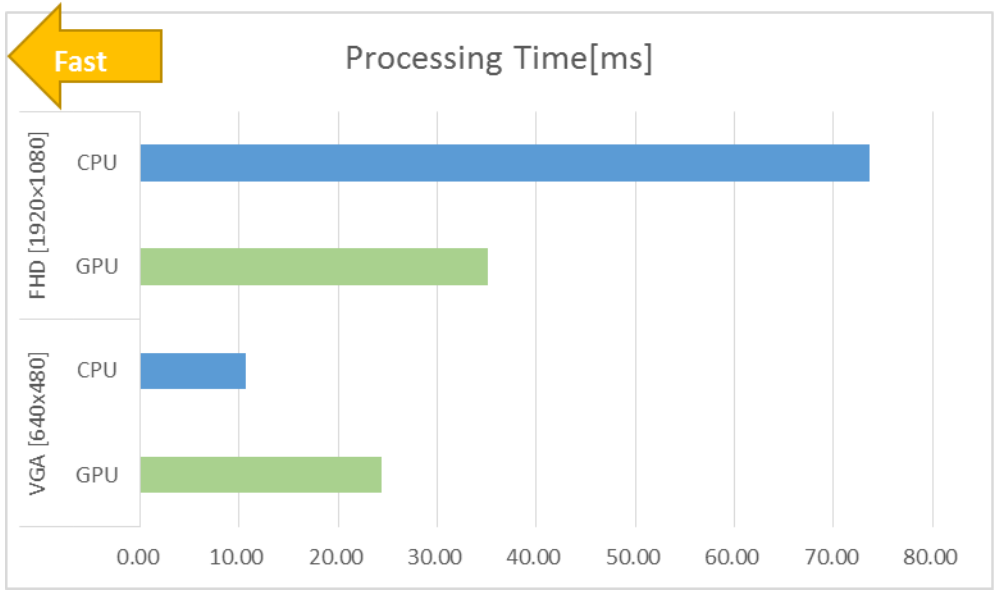
Performance counter stats for 'TestOpenCL_threshold --cl=true ,,,

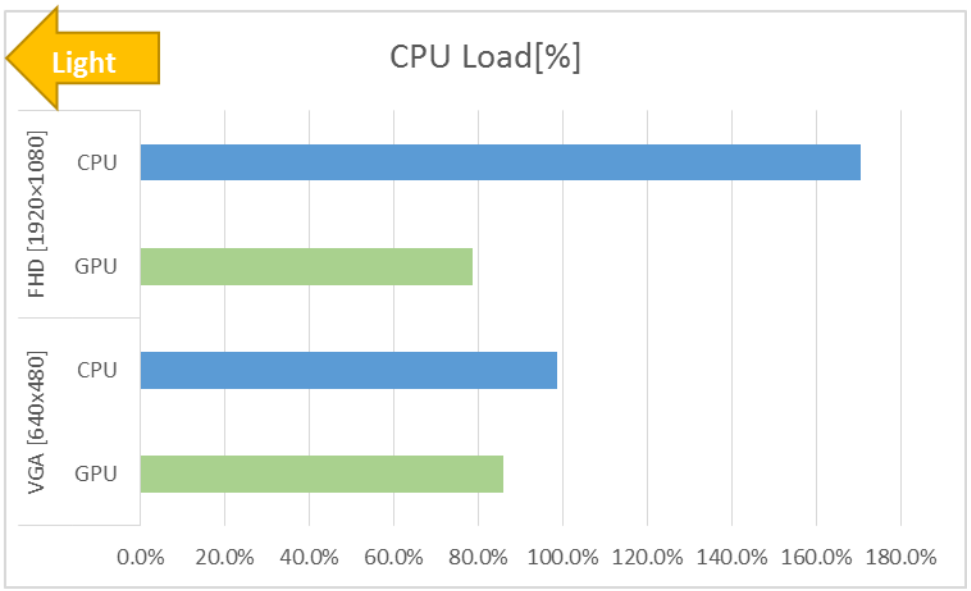
174.865064	task-clock (msec)	# 0.701 CPUs utilized
304	context-switches	# 0.002 M/sec
1	cpu-migrations	# 0.006 K/sec
3771	page-faults	# 0.022 M/sec
261558114	cycles	# 1.496 GHz
288810600	instructions	# 1.10 insn per cycle
<not supported>	branches	
1558031	branch-misses	# 0.00% of all branches

NTT DATA MSE Corporation

Performance Verification (5/5)

Resize image(Bi-linear interpolation) : resize(0.5)





Longer processing time but lower CPU load by "99%->86%" in VGA.
Shorter processing time by "74ms->35ms" and lower CPU load by "170%->78%" in Full-HD and GPU effects work for both!

Snap shot）Case CPU／ Full-HD
Performance counter stats for 'TestOpenCL_resize --cl=false ,,,

304.999219	task-clock (msec)	#	1.701 CPUs utilized
1300	context-switches	#	0.004 M/sec
17	cpu-migrations	#	0.056 K/sec
2353	page-faults	#	0.008 M/sec
454090850	cycles	#	1.489 GHz
315853627	instructions	#	0.70 insn per cycle
<not supported>	branches		
1495224	branch-misses	#	0.00% of all branches

Snap shot）Case GPU／ Full-HD
Performance counter stats for 'TestOpenCL_resize --cl=true ,,,

159.925113	task-clock (msec)	#	0.753 CPUs utilized
305	context-switches	#	0.002 M/sec
1	cpu-migrations	#	0.006 K/sec
3735	page-faults	#	0.023 M/sec
239104934	cycles	#	1.495 GHz
254117677	instructions	#	1.06 insn per cycle
<not supported>	branches		
1716419	branch-misses	#	0.00% of all branches

NTT DATA MSE Corporation

Consideration

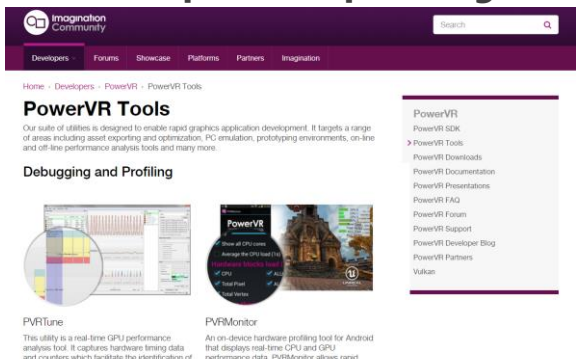
■ Verified OpenCL(GPU) supporting effects in R-Car H3 / AGL

- Effect to shorten the processing time and lower CPU load **according to process contents (adaptable algorithm)**
- The **larger the image size (calculation amount)** is, the more the effect works (Because the image size and operating volume are small due to 100x100[pix]/8bit for image processing in pre-developed demo, it is suggested that the copy overhead to device array and the increase in CPU stall have a stronger control.)

■ Next Step

- Considered that it is important to determine the cases and process contents applied by OpenCL (adaptable algorithm and image size)
- Perform the profiling and tuning (e.g. what causes CPU[ARM] stall? how is it improved?) in order to utilize CPU/GPU features at most

For example, GPU profiling



PVRTune provided by Imagination. See also:
https://community.imgtec.com/developers/powervr/tools/

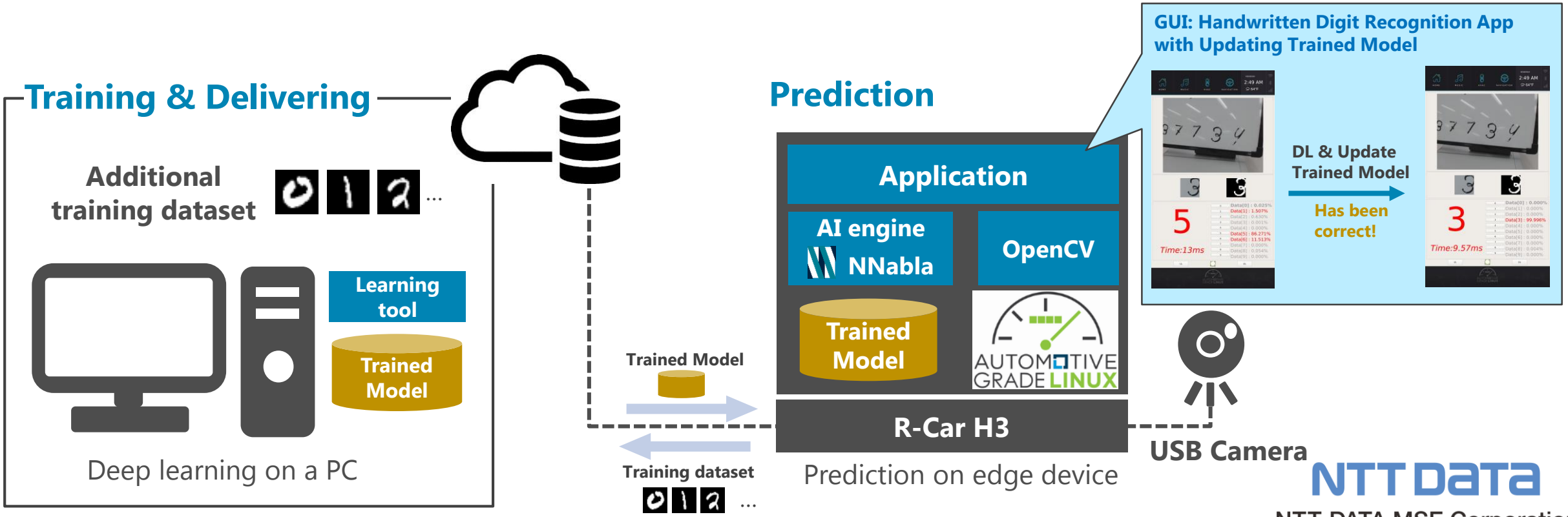
Possibility verification of continuous improvement of inference accuracy for edge AI

NTT DATA MSE Corporation

System Overview

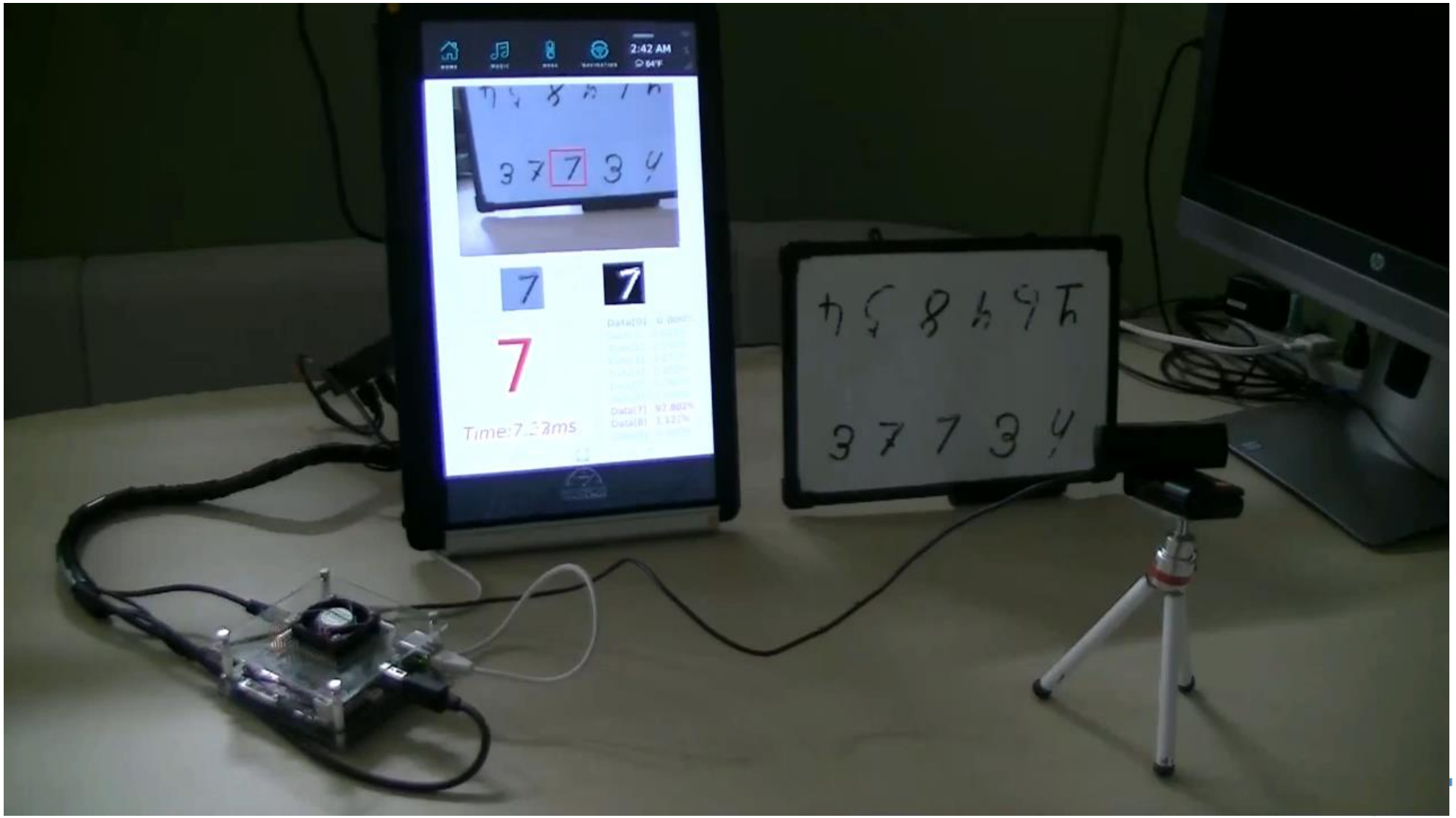
Mechanism to enable continuous improvement of inference accuracy for edge AI

- Training dataset upload (unrecognized / low recognition rate) from edge device
- Additional training on PC via the cloud
- Pre-trained models delivery to edge device and updates



NTT DATA MSE Corporation

Demonstration & Verification Result



NTT DATA MSE Corporation

Demonstration & Verification Result

Found the possibility of **continuous improvement of inference accuracy** for edge AI

- Improved the recognition result
- For us, available to build new trained models with a little effort and time because of only additional data training (if the pre-trained models are in place)

  First training (60000 samples): 90 sec.

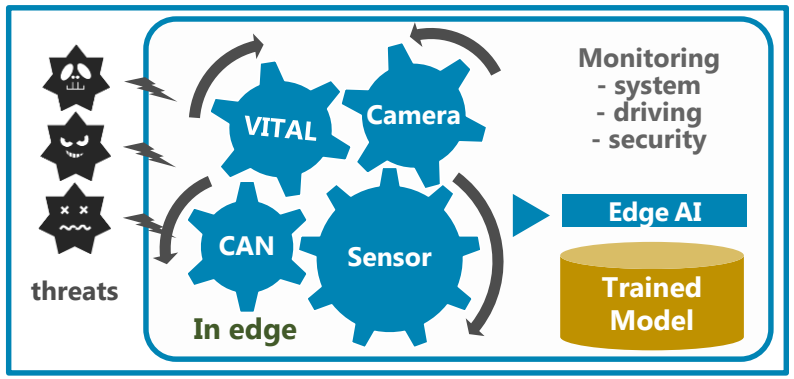
  Additional training (e.g. 13 samples): 4 sec.
- Maintained a generalization accuracy (in-house verification: about 99 %)
  - ✓ Low possibility of incorrect recognition of correctly pre-recognized data.
  - ✓ Also, high possibility of correct recognition of unknown input data.
- Enable to improve the inference results only by updating the pre-trained model (just one .npp file) on edge device
  - ✓ Expect to perform "Recognition accuracy improvement" in the background

Demonstration & Verification Result

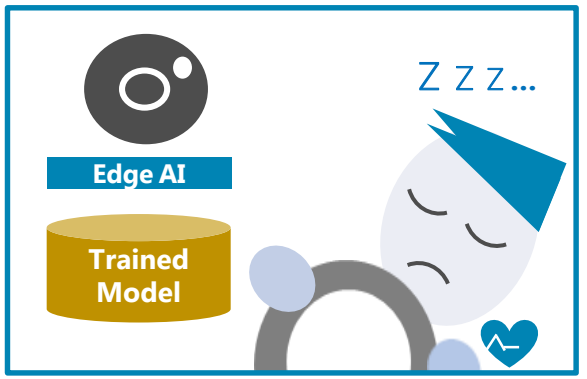
Next Step

- What types of **practical use cases** are available?

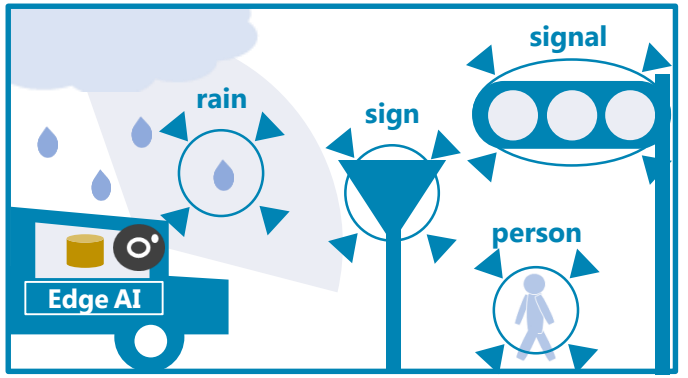
Future use-cases



Use various information of embedded devices



Detect driver's drowsiness



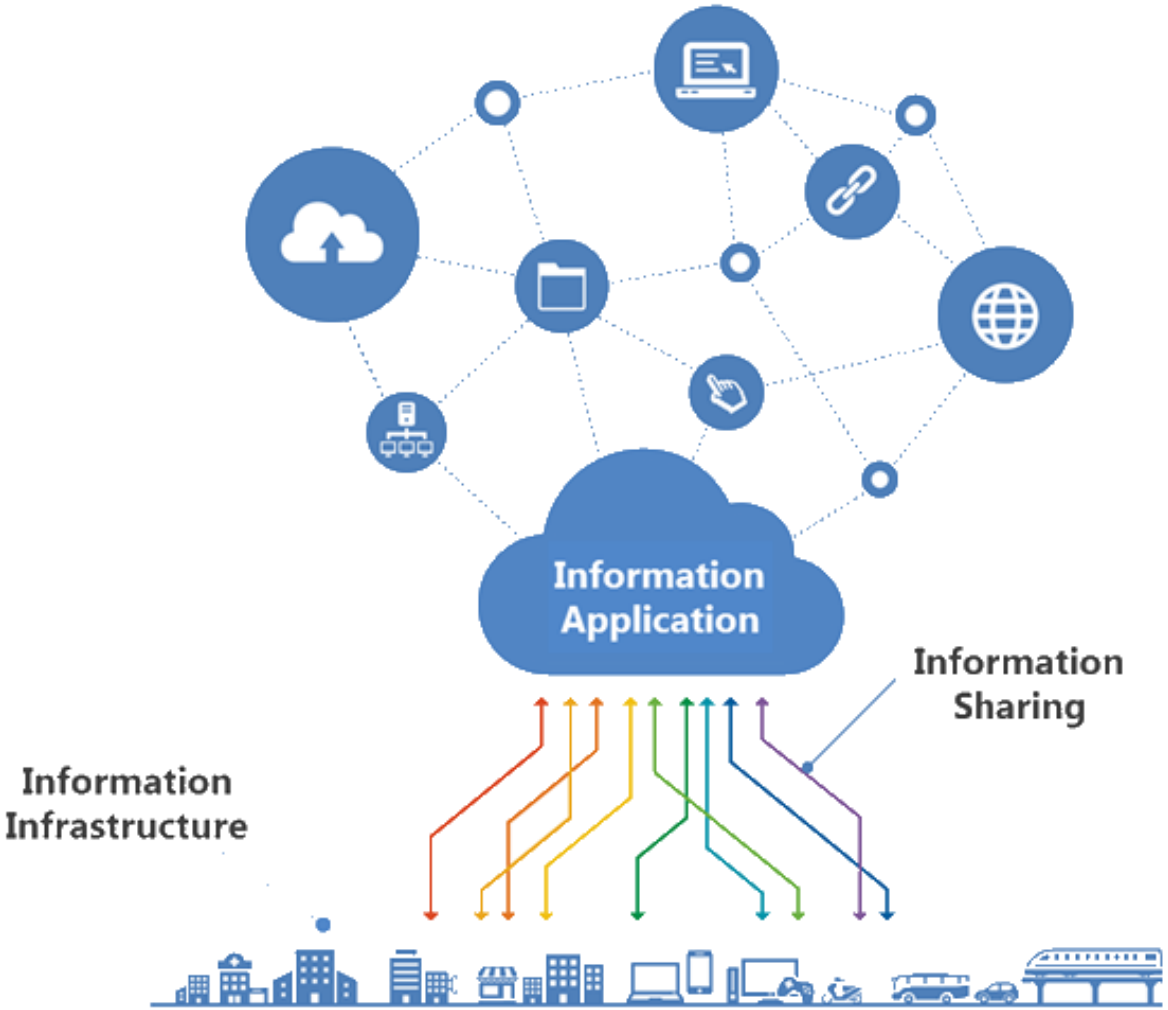
Improvement of autonomous driving technology

NTT DATA MSE Corporation

Questions?

NTT DATA MSE Corporation

Thank you very much!!



Smart Life Community®

NTT DATA

NTT DATA MSE Corporation