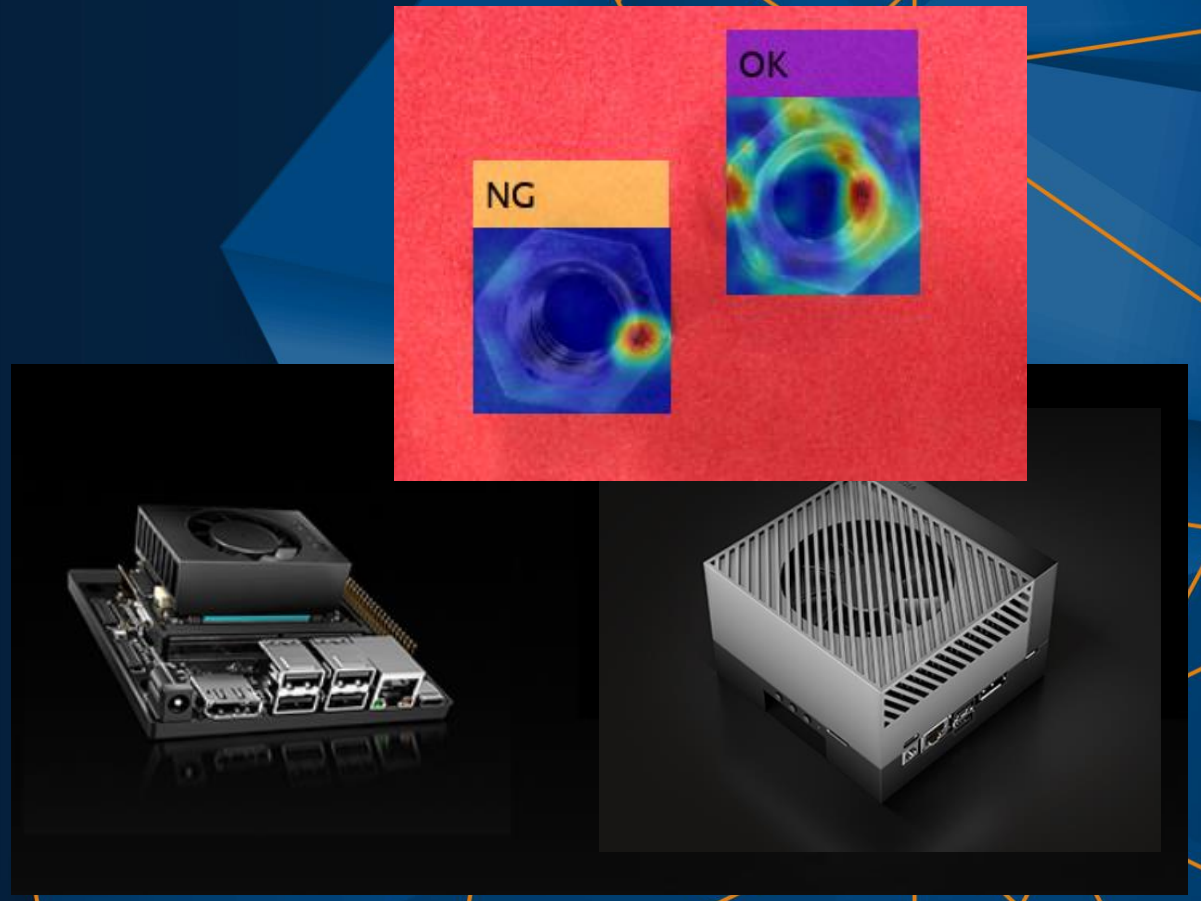
MATLAB EXPO

MATLAB과 AGX Jetson Orin을 이용한 Edge AI 솔루션 데모

신행재 부장 & 김종남 부장



MathWorks

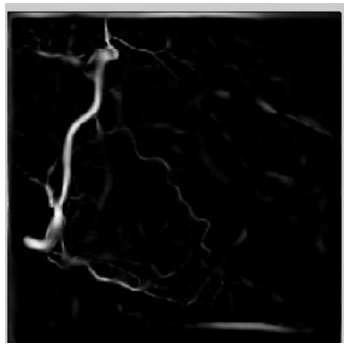
GPU Coder for Image Processing and Computer Vision

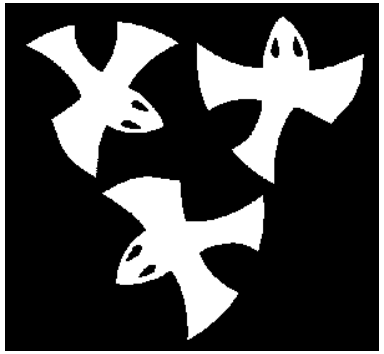
Fog removal

5x speedup

Frangi filter

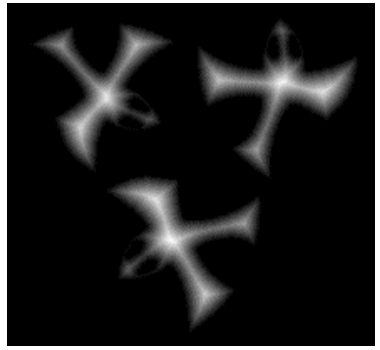
3x speedup





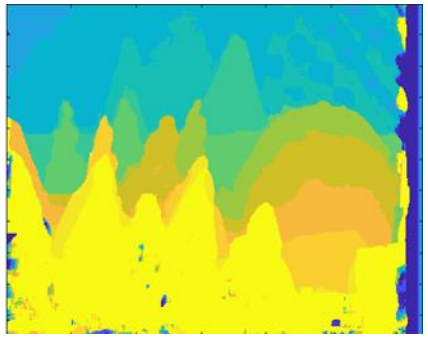
Distance transform

8x speedup



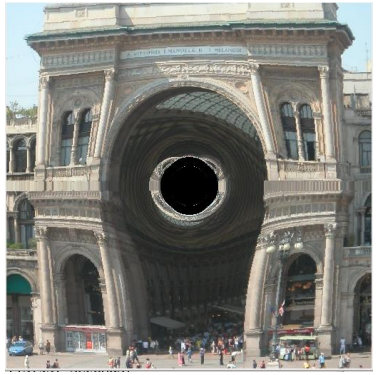
Stereo disparity

50x speedup



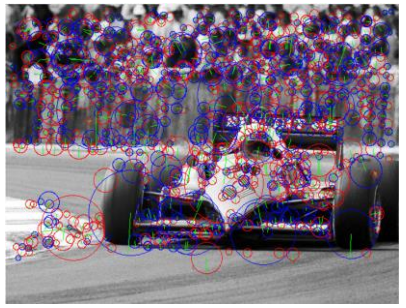
Ray tracing

18x speedup



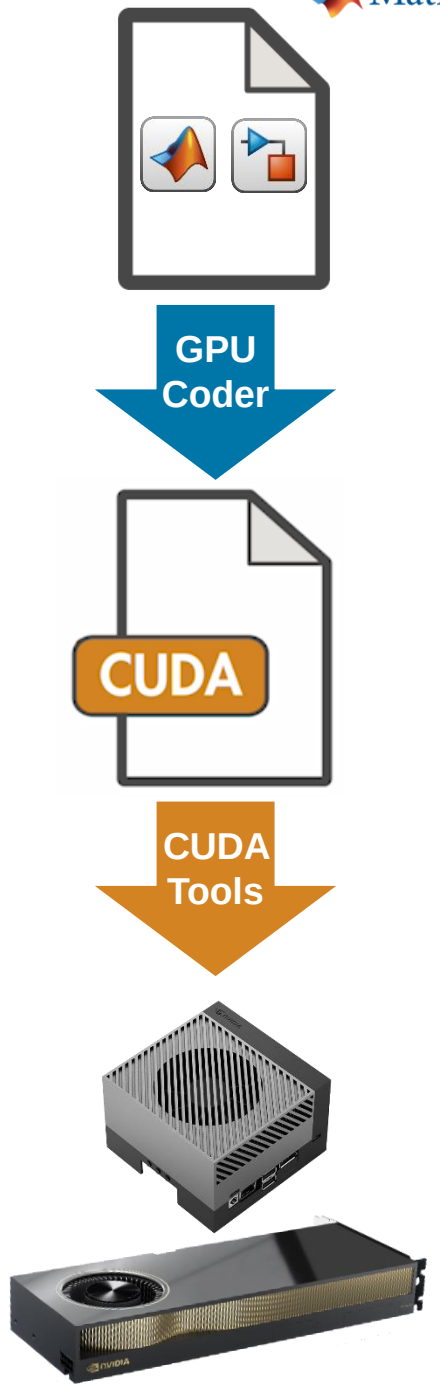
SURF feature extraction

700x speedup

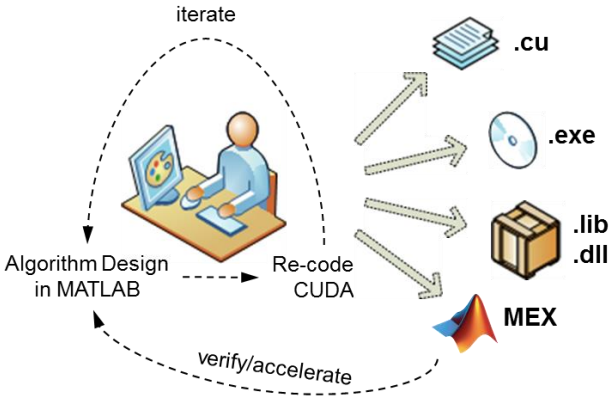


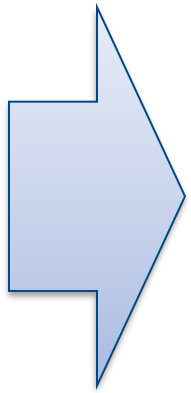
GPU Coder

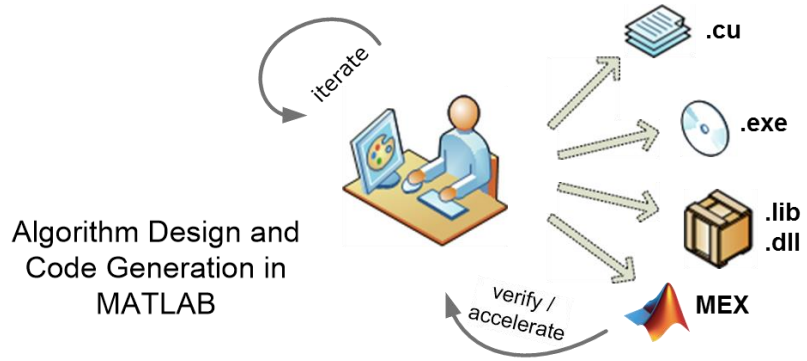
- Generate optimized CUDA code from MATLAB and Simulink for deep learning, embedded vision, and autonomous systems
- Integrate generated code as source code and static or dynamic libraries
- Generated CUDA is portable across NVIDIA GPUs
- Generated code calls optimized NVIDIA CUDA libraries, including TensorRT, cuDNN, cuSolver, cuFFT, and cuBLAS
- Incorporate handwritten CUDA code into MATLAB algorithms and generated code
- Prototype algorithms on modern GPUs including the Titan RTX and Jetson AGX Orin
- Accelerate computationally intensive portions of your MATLAB code and Simulink models using generated CUDA code



Why Use GPU Coder?







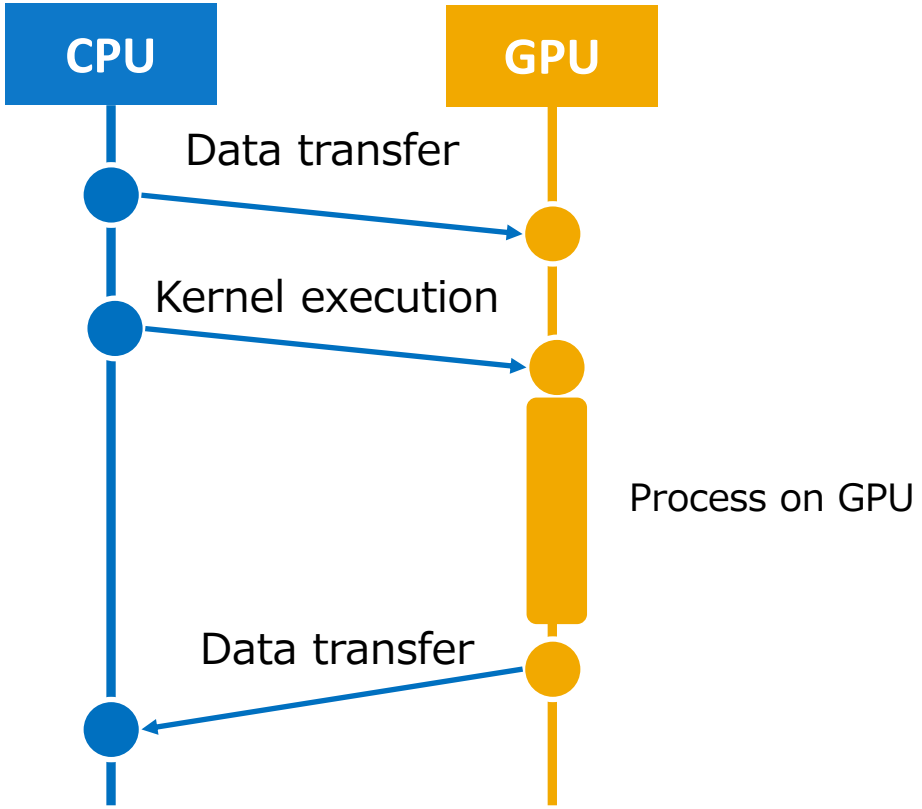
Pains: Hand code

- **Cannot code in CUDA**
- Time consuming
- Manual Coding Errors
- Multiple implementations
- Expensive

Solution: GPU Coder

- Automatically convert to CUDA
- Get to CUDA faster
- Eliminate manual coding errors
- Maintain Single “Truth”
- Stay within MATLAB & Simulink at a higher level

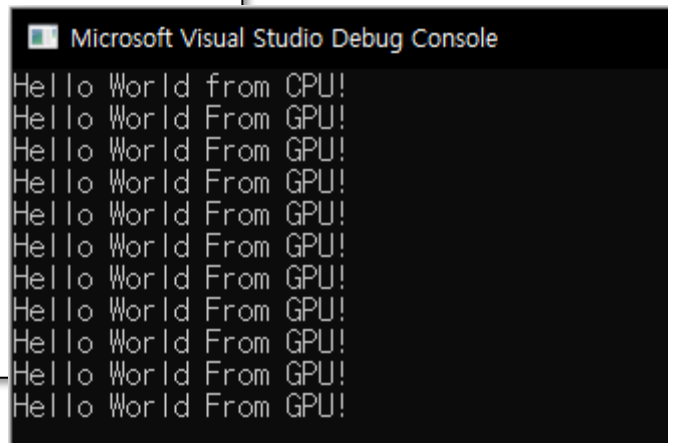
Run Hello World on GPU




```
__global__ void helloFromGPU()
{
    printf("Hello World from GPU!\n");
}

int main(int argc, char **argv)
{
    printf("Hello World from CPU!\n");

    helloFromGPU<<<1, 10>>>();
    return 0;
}
```



・Kernel call(special syntax)

```
kernelFunc<<<Block_dim, Thread_dim>>>(a, b, c);
```

For example, if you could do this ...

Linear Algebra routine, SAXPY example

Scalarized MATLAB

```
for i = 1:length(x)
    z(i) = a .* x(i) + y(i);
end
```

Vectorized MATLAB

```
z = a .* x + y;
```

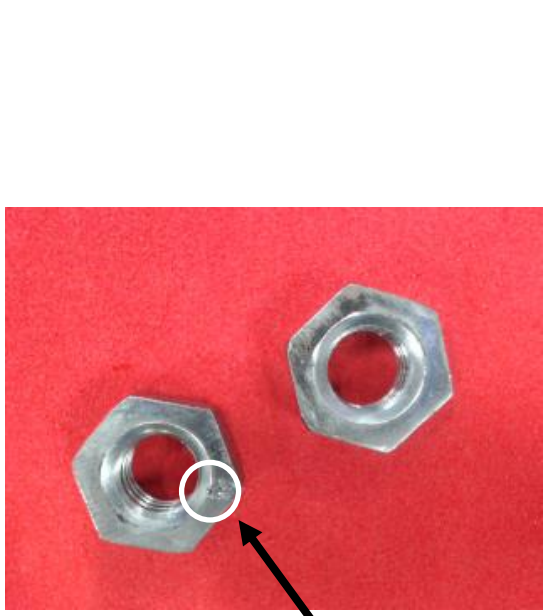
GPU Coder

```
static __global__ __launch_bounds__(512, 1) void saxpy_kernel1(const real32_T *y,
  const real32_T *x, real32_T a, real_T *z)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (!(i >= 1048576)) {
    z[i] = (real_T)(a * x[i] + y[i]);
  }
}

void saxpy(real32_T a, const real32_T x[1048576], const real32_T y[1048576],
           real_T z[1048576])
{
  real32_T *gpu_y;
  real32_T *gpu_x;
  real_T *gpu_z;
  cudaMalloc(&gpu_z, 8388608UL);
  cudaMalloc(&gpu_x, 4194304UL);
  cudaMalloc(&gpu_y, 4194304UL);
  cudaMemcpy((void *)gpu_y, (void *)&y[0], 4194304UL, cudaMemcpyHostToDevice);
  cudaMemcpy((void *)gpu_x, (void *)&x[0], 4194304UL, cudaMemcpyHostToDevice);
  saxpy_kernel1<<<dim3(2048U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_y, gpu_x, a,
    gpu_z);
  cudaMemcpy((void *)&z[0], (void *)gpu_z, 8388608UL, cudaMemcpyDeviceToHost);
  cudaFree(gpu_y);
  cudaFree(gpu_x);
  cudaFree(gpu_z);
}
```

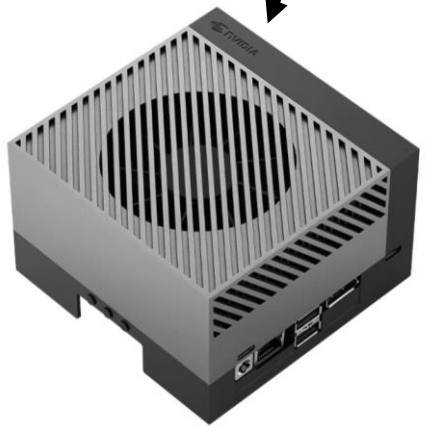
Automatic compilation from a highly extensible language to a high performance language

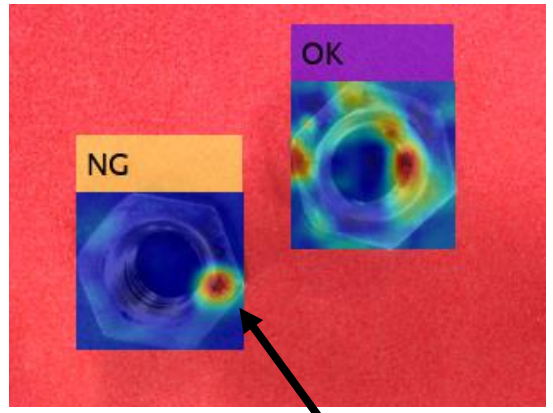
Implementation of pass / fail judgment algorithm by deep learning



Defective hexagon nut
(Scratches on the surface, etc.)

NVIDIA Jetson Orin

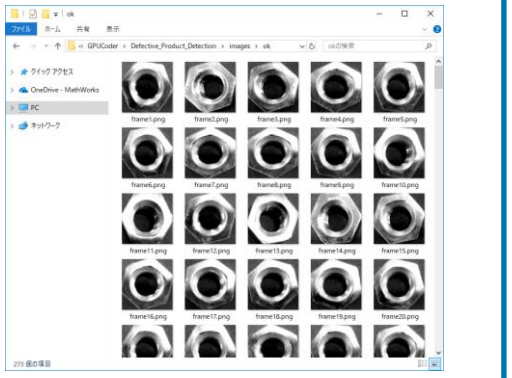




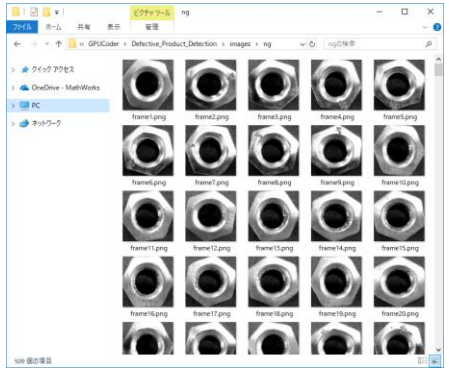
With Class Activation Map
Visualize the area of interest

Exercise 8: Creating a CNN for Pass / Fail Judgment by Transfer Learning-Data Preparation

- Handle image data using imageDatastore
  - Work with train_SqueezeNet.m



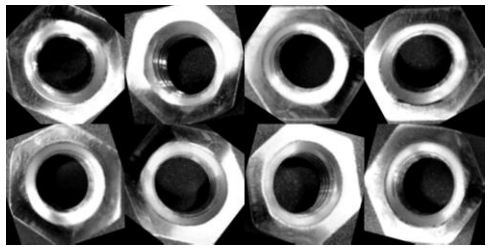
OK images



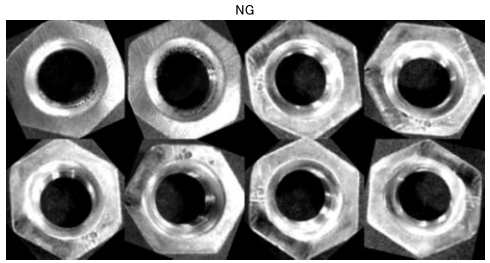
NG images

The original image

In a random manner
Rotate and flip



OK images

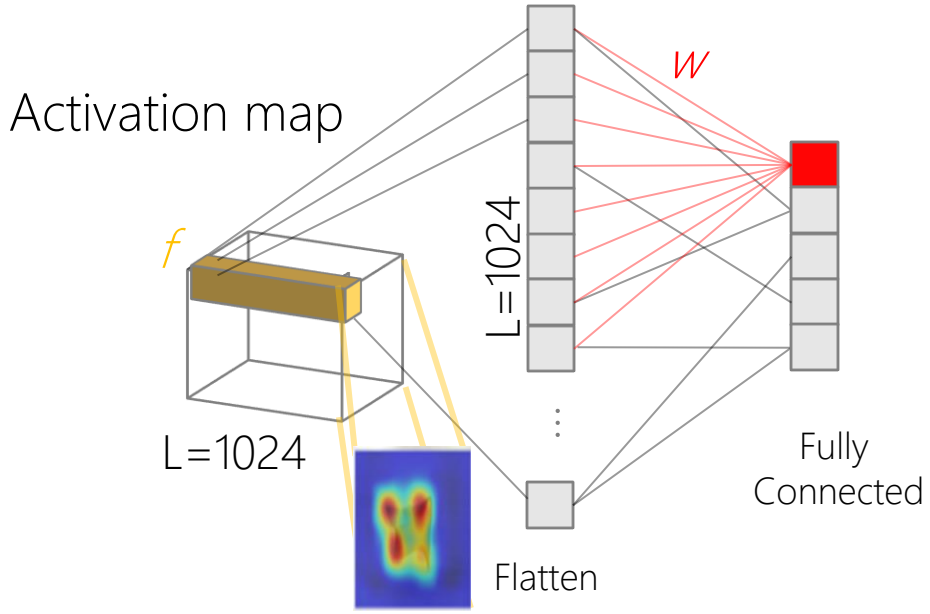


NG images

Augmentation

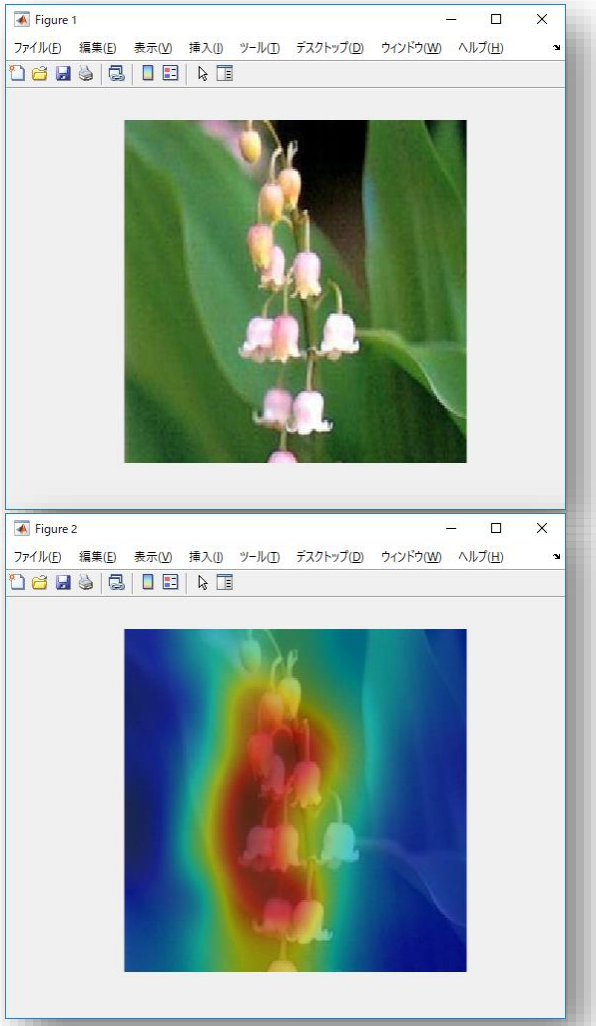
- Divide the data for training and validation
  - Modify line 12
- Randomly rotate the image to increase the number of training data
  - Modify line 15 and add random rotation instructions
- **Check the added data to see if the option you added is enabled**
  - Use the augment function

About CAM (Class Activation Mapping)



$$\sum_k f_k w_k = f_1 w_1 + f_2 w_2 + \ldots + f_{1024} w_{1024}$$

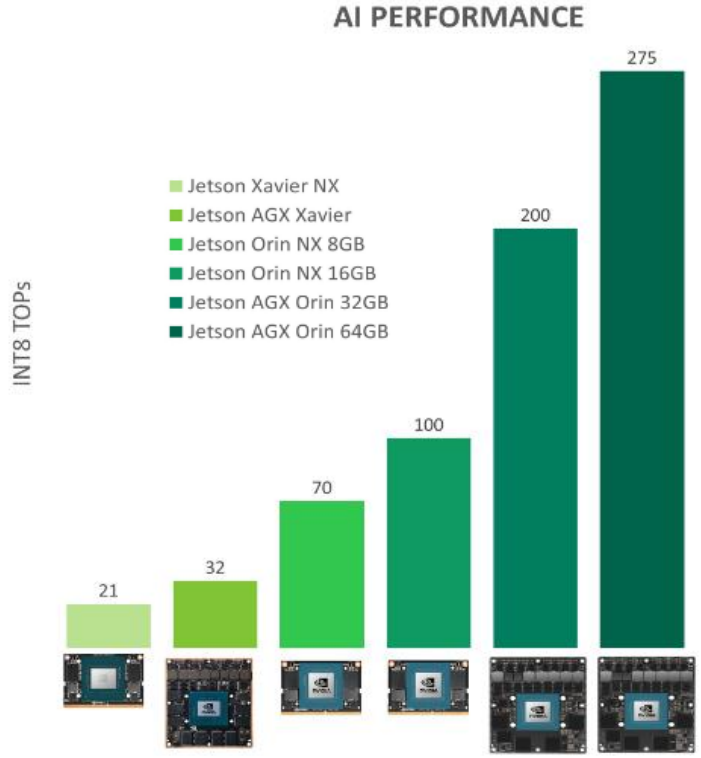
```
dotProduct =
bsxfun(@times,imageActivations,weightVector);
classActivationMap = sum(dotProduct,3);
```

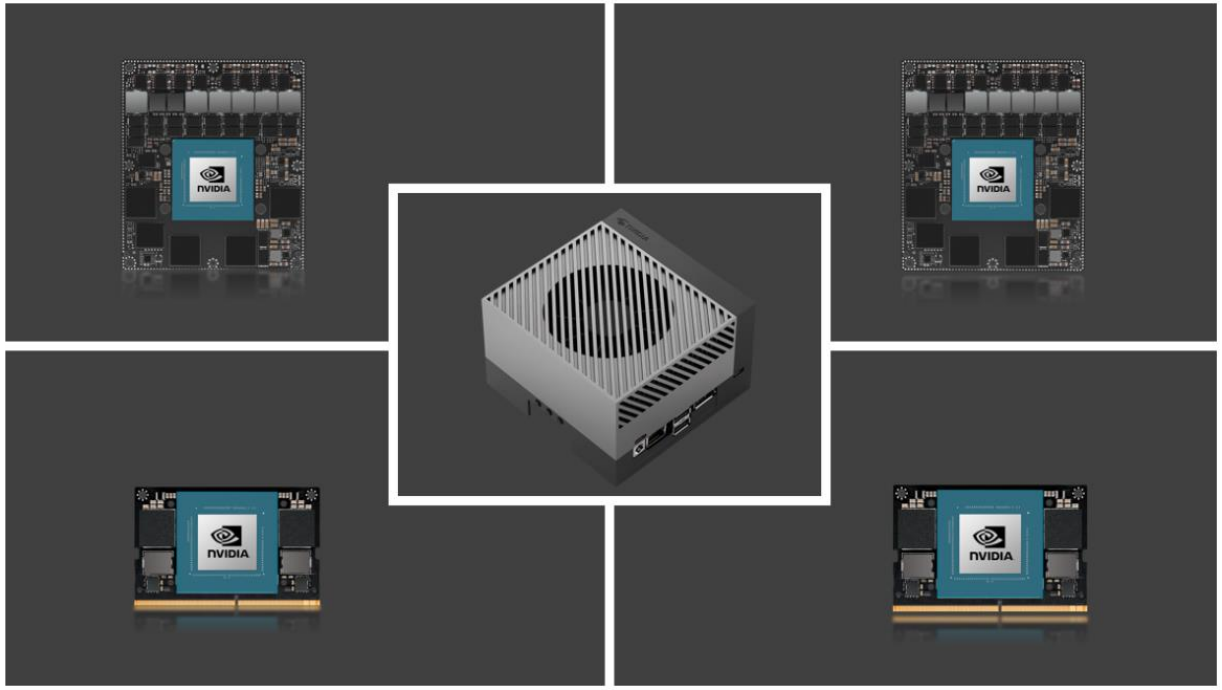


Visualize Class Activation Map

https://jp.mathworks.com/help/releases/R2019a/deeplearning/examples/investigate-network-predictions-using-class-activation-mapping.html

Delivering Server-Class Performance at the Edge with NVIDIA Jetson Orin



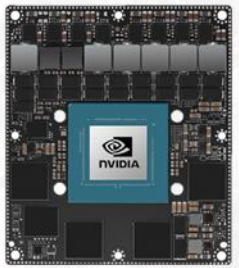


	Jetson AGX Orin series			Jetson Orin NX series		Jetson Orin Nano series		
	Jetson AGX Orin Developer Kit	Jetson AGX Orin 64GB	Jetson AGX Orin 32GB	Jetson Orin NX 16GB	Jetson Orin NX 8GB	Jetson Orin Nano Developer Kit	Jetson Orin Nano 8GB	Jetson Orin Nano 4GB
AI Performance	275 TOPS		200 TOPS	100 TOPS	70 TOPS	40 TOPS		20 TOPS
GPU	2048-core NVIDIA Ampere architecture GPU with 64 Tensor Cores		1792-core NVIDIA Ampere architecture GPU with 56 Tensor Cores	1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores		1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores		512-core NVIDIA Ampere architecture GPU with 16 Tensor Cores
GPU Max Frequency	1.3 GHz		930 MHz	918 MHz	765 MHz	625 MHz		

https://developer.nvidia.com/blog/delivering-server-class-performance-at-the-edge-with-nvidia-jetson-orin/

*TOPS: Tera Operations Per Second

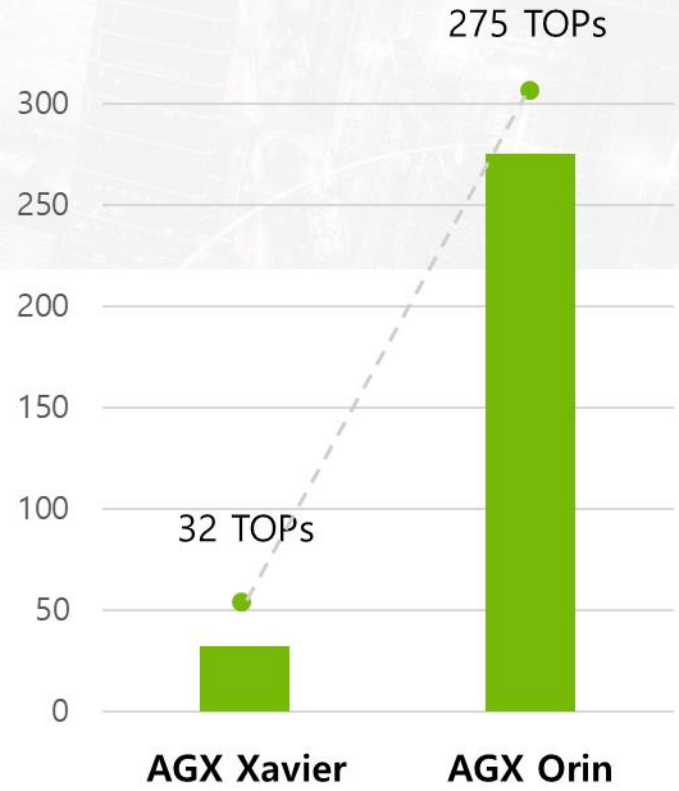
Jetson AGX Xavier vs Jetson AGX Orin Migration



Specification	Jetson AGX Xavier (64GB)	Jetson AGX Orin (64GB)
AI Performance (INT8)	**32 TOPs**	**275 TOPs**
GPU	**Volta** 512 CUDA Core | 64 Tensor Core	**Ampere** 2048 CUDA Core | 64 Tensor Core
CPU	**NVIDIA Carmel ARMv8.2** 8 Core	**Arm Cortex A78AE v8.2** 12 Core
Memory	**64GB 256-Bit** **LPDDR4x | 137GB/s**	**64GB 256-Bit** **LPDDR5 | 204GB/s**
DL Accelerator	**NVDLA v1.0 x 2**	**NVDLA v2.0 x 2**
Vision Accelerator	**PVA v1.0 x 2**	**PVA v2.0 x 1**

AI Performance 성능

최대 8배



Jetson AGX Xavier vs Jetson AGX Orin Migration

Specification		Jetson AGX Xavier	Jetson AGX Orin
Networking	RGMII	10/100/1000 Mbit	10/100/1000 Mbit
	MGBE	Not Supported	4 x 10Gbe XFI
Video Input	CSI	16 lanes MIPI CSI-2 D-PHY 1.2 (40 Gbps) C-PHY 1.1 (62 Gbps)	16 lanes MIPI CSI-2 D-PHY 2.1 (40 Gbps) C-PHY 2.0 (164 Gbps)
	SLVS	8-lane	Not Supported
USB 3.x		3.1 Gen 2(10Gbps) x 3	3.2 Gen 2(10Gbps) x 3
PCIe		PCIe Gen 4 ➔ 2x1 / 1x2 / 1x4 / 1x8 All Lane Support Root Port. Only 1x8 has RP and EP support. * 1x8 shared with SLVS	PCIe Gen 4 ➔ 2x1 / 1x4 / 2x8 All Lane Support Root Port. Only 2x8 has RP and EP support. * '4lane of 1x8' shared with MGBE
Display		3 Multi-mode (2x4k60) (e)DP 1.4 HDMI 2.0a	1 Multi-mode (8k60, 2x4K60) (e)DP 1.4 (HBR3, MST, DSCT) HDMI 2.1
Audio	I2S	x4	x4(compatible) + x2(additional)
	DMIC	x2	x2(compatible) + x2(additional)
	DSPK	x1	x1(compatible) + x1(additional)
UART		x5	x4(compatible) + x2(additional)
SPI		x3	x3(compatible) + x1(additional)