MATLAB EXPO

엣지 AI기술을 위한 딥러닝 모델의 1-bit 양자화

김정훈, 네패스

MathWorks

목차

발표자 김정훈

- 고려대학교 전기전자공학 석사
- (주)네패스 딥러닝 엔지니어 전문연구요원
- Contact: jeonghoon.samuel@gmail.com
- 관심 키워드: Neural Network Quantization, Compact Network Design, Robotics Perceptions, State Estimation, …
- 2019 활동 사항:
  - Google Developer Group Gwangju DevFest 2019 발표자
  - 한국 기술 교육 대학교 온라인 평생 교육원 자문
  - 삼성 오픈 소스 컨퍼런스 (SOSCON) 2019 심사위원
  - Mathworks Advisory Board 2019
  - 인공지능 로보틱스 커뮤니티 운영자
  - Neural Network Quantization & Compact Network Development 스터디 리더

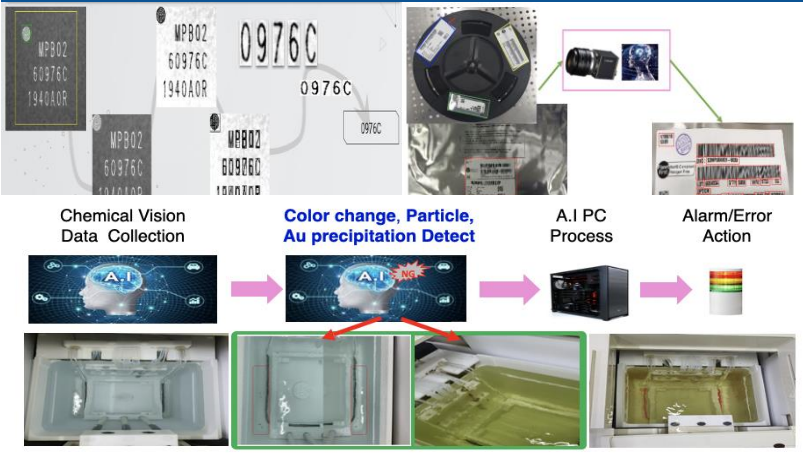
함께 연구하고, 나누며, 성장하는데에 큰 보람을 느끼고 있습니다!

오늘의 발표 내용: Binarized Neural Networks on MATLAB

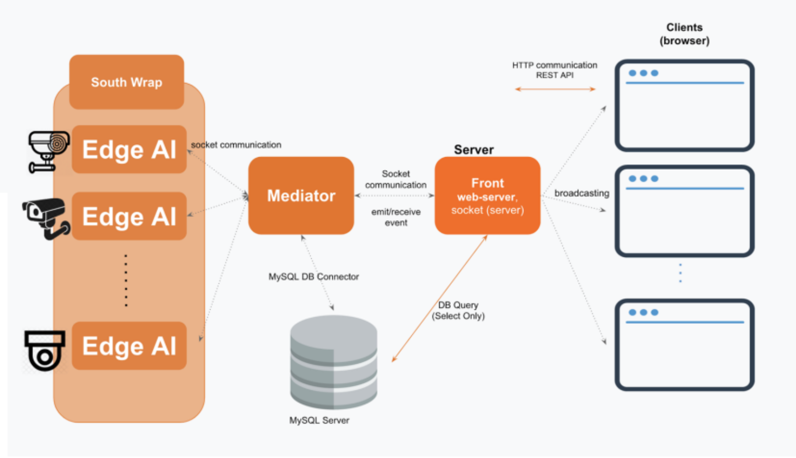
nepes Future Intelligence

nepes

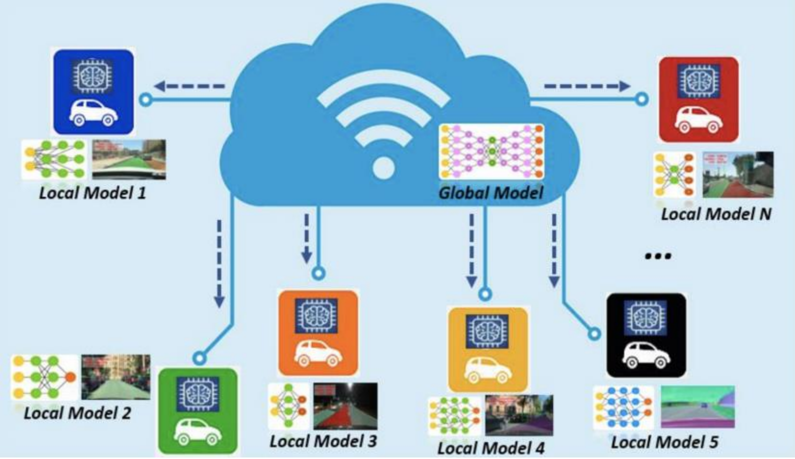
AI Factory



AI CCTV System



Research Projects



Device	Solution
Hardware	Software

Real Applications

&

Core Value Research

프로젝트 개요: 매트랩을 통한 이진화 신경망 학습기 개발

Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to $+1$ or -1

Matthieu Courbariaux*[1] MATTHIEU.COURBARIAUX@GMAIL.COM
Itay Hubara*[2] ITAYHUBARA@GMAIL.COM
Daniel Soudry[3] DANIEL.SOUDRY@GMAIL.COM
Ran El-Yaniv[2] RANI@CS.TECHNION.AC.IL
Yoshua Bengio[1,4] YOSHUA.UMONTREAL@GMAIL.COM
[1]Université de Montréal
[2]Technion - Israel Institute of Technology
[3]Columbia University
[4]CIFAR Senior Fellow
*Indicates equal contribution. Ordering determined by coin flip.

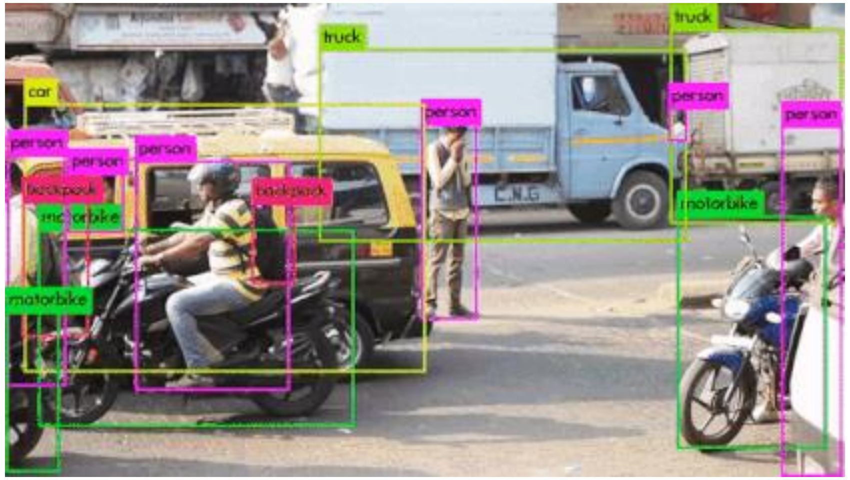
MATLAB

MATLAB은 제공해주는 기능만 사용 할 수 있는 폐쇄적 개발환경인가?
이 논문을 MATLAB으로 어떻게 구현할 수 있을까?

Hubara, Itay, et al. "Binarized neural networks." Advances in neural information processing systems. 2016.

개발 배경

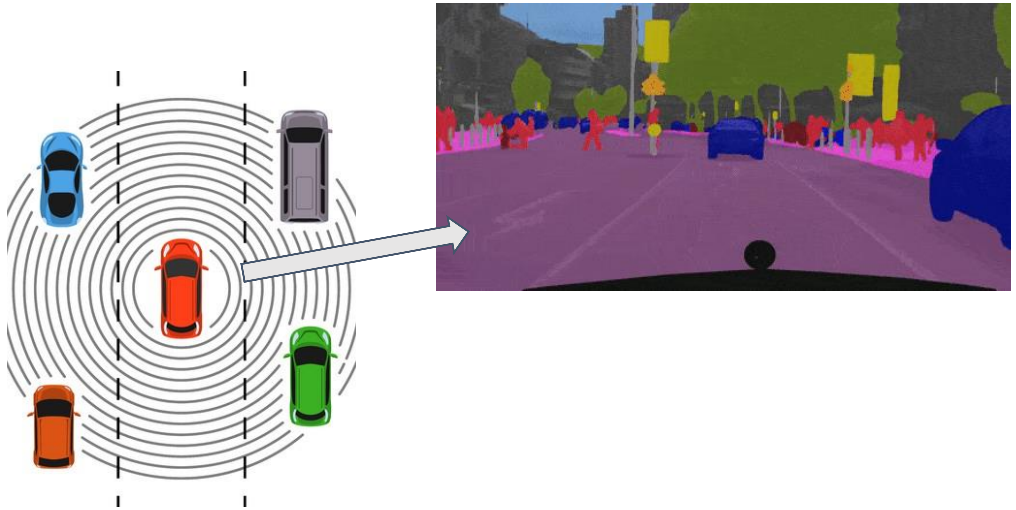
훌륭한 성능을 내는 신경망 기반 어플리케이션들



Object Detection

Semantic Segmentation

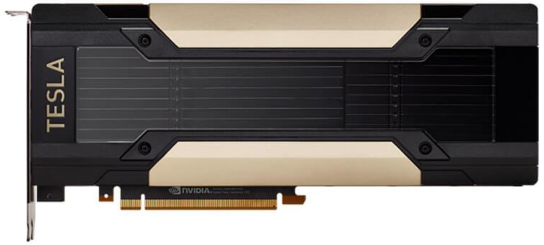
개발 배경



Real Time Performance

Customer

개발 배경

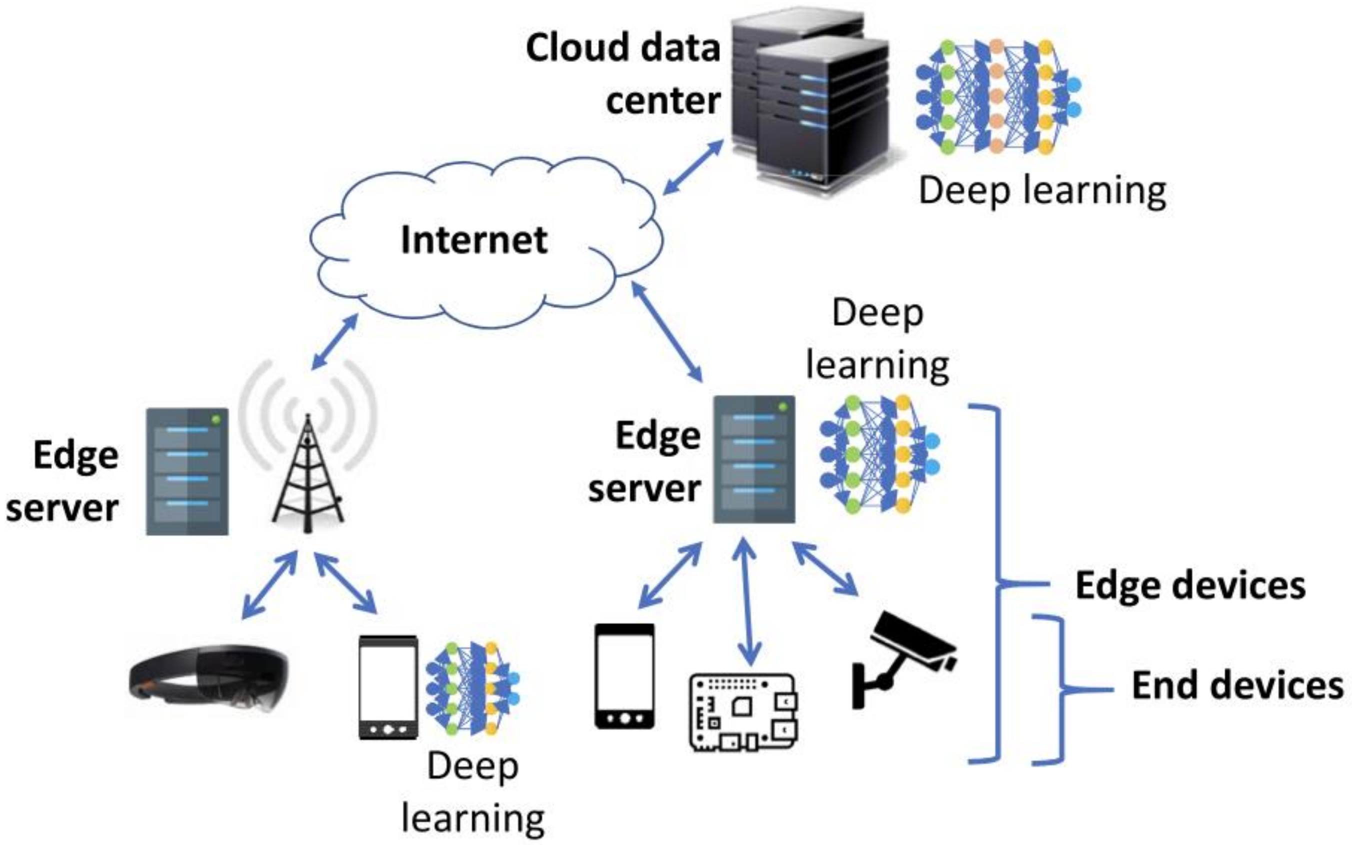


x4

On a High-end GPU

Customer

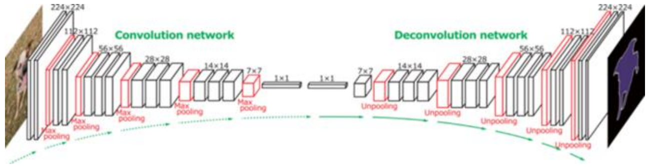
개발 배경



Various Inference Environments of Deep Learning Network

Chen, Jiasi, and Xukan Ran. "Deep learning with edge computing: A review." Proceedings of the IEEE 107.8 (2019): 1655-1674.

개발 배경

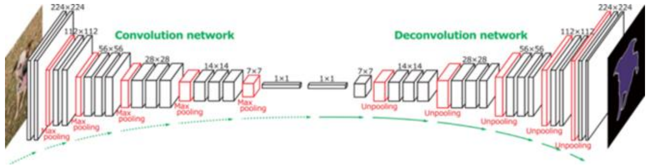


Neural Network

?

Target Environment

개발 배경



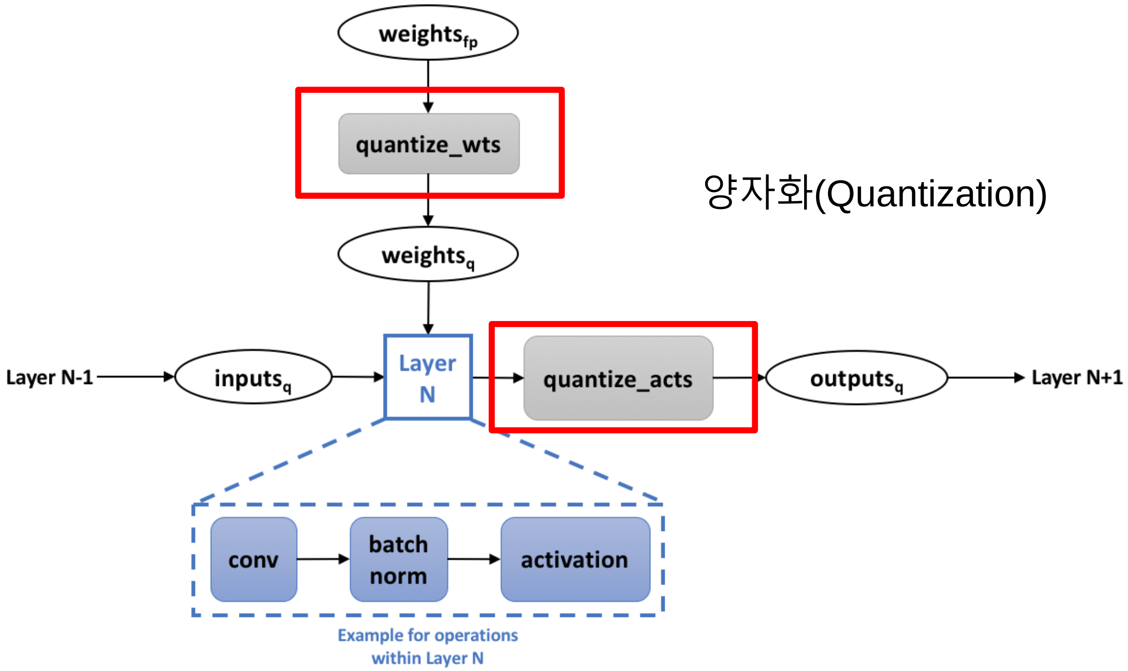
Neural Network

Neural Network Quantization

Target Environment

기술적인 해결과제

네트워크를 구성하는 weight와 activation의 bit를 낮추어 네트워크를 **경량화** 하거나 연산을 **가속**



양자화(Quantization)

32-bit (Full Precision)

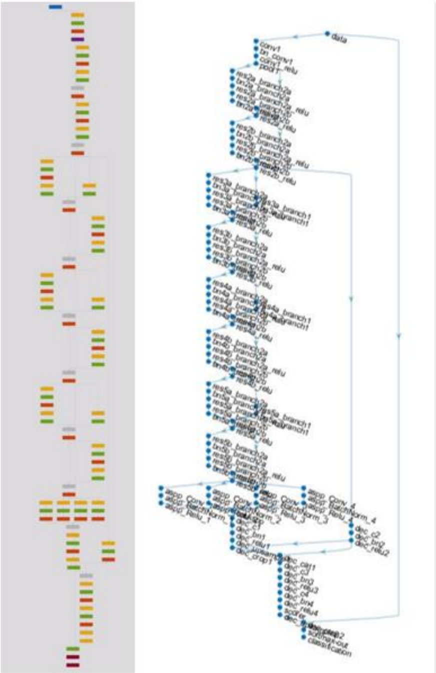
⇩

1-bit Binarization

MATLAB에서 **1-bit quantization에 대하여 구현해보기**!
(현재 MATLAB 2020a에서 8-bit quantization 기능을 일부 지원하고 있습니다. 오늘 바로 사용해보세요!)

https://nervanasystems.github.io/distiller/quantization.html

기술적인 해결과제



DeepLabV3+ (Baseline: ResNet18)

Width	Height	Input Channel	Output Channel	Total
7	7	3	64	9408
3	3	64	64	36864
3	3	64	64	36864
3	3	64	64	36864
3	3	64	64	36864
1	1	64	48	3072
1	1	64	128	8192
3	3	64	128	73728
3	3	128	128	147456
3	3	128	128	147456
3	3	128	128	147456
3	3	128	256	294912
1	1	128	256	32768
3	3	256	256	589824
3	3	256	256	589824
3	3	256	256	589824
1	1	256	512	131072
3	3	256	512	1179648
3	3	512	512	2359296
3	3	512	512	2359296
3	3	512	512	2359296
3	3	512	256	1179648
3	3	512	256	1179648
3	3	512	256	1179648
1	1	512	256	131072
1	1	1024	256	262144
8	8	256	256	4194304
3	3	304	256	700416
3	3	256	256	589824
1	1	256	11	2816
8	8	11	11	7744

Architecture Weight Count

만약 DeepLabV3+ 를 1-bit Quantization한다면?

FP32 vs 1 Bit

Total Weight Count

	bit	byte	
FP32	659111936	82388992	82.388992MB
1bit	20597248	2574656	2.574656 MB

Maximum Activation (Input Size: 360 x 480 x 3)

	bit	byte	
FP32	105,062,400	13132800	13.1328 MB
1bit	3,283,200	172800	0.1728 MB

당연한 결과지만, 32-bit와 1-bit는 Volume에서 32배 차이!

기술적인 해결과제

Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to $+1$ or -1

Matthieu Courbariaux*[1] MATTHIEU.COURBARIAUX@GMAIL.COM
Itay Hubara*[2] ITAYHUBARA@GMAIL.COM
Daniel Soudry[3] DANIEL.SOUDRY@GMAIL.COM
Ran El-Yaniv[2] RANI@CS.TECHNION.AC.IL
Yoshua Bengio[1,4] YOSHUA.UMONTREAL@GMAIL.COM
[1]Université de Montréal
[2]Technion - Israel Institute of Technology
[3]Columbia University
[4]CIFAR Senior Fellow
*Indicates equal contribution. Ordering determined by coin flip.

Check point:

1. Binarization function
2. Backpropagation of BNNs (Algorithm1)
3. Computing gradients and updates

Hubara, Itay, et al. "Binarized neural networks." Advances in neural information processing systems. 2016.

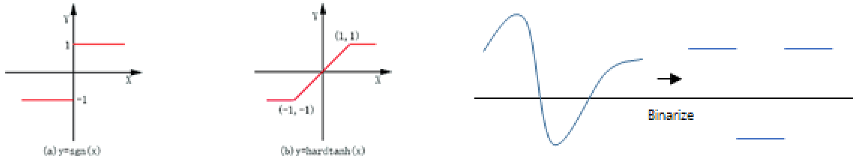
기술적인 해결과제 : BNN 구현

Binarization

Deterministic Binarization

$$x^b = \mathrm{Sign}(x) = \begin{cases} +1 & \text{if } x \geq 0, \\ -1 & \text{otherwise,} \end{cases} \qquad (1)$$

where x^b is the binarized variable (weight or activation) and x the real-valued variable



1.12	3.42	-1.5	-12
32	-1	-5	15
24	0.55	-54	0.24
-0.1	0.1	-0.2	2

→

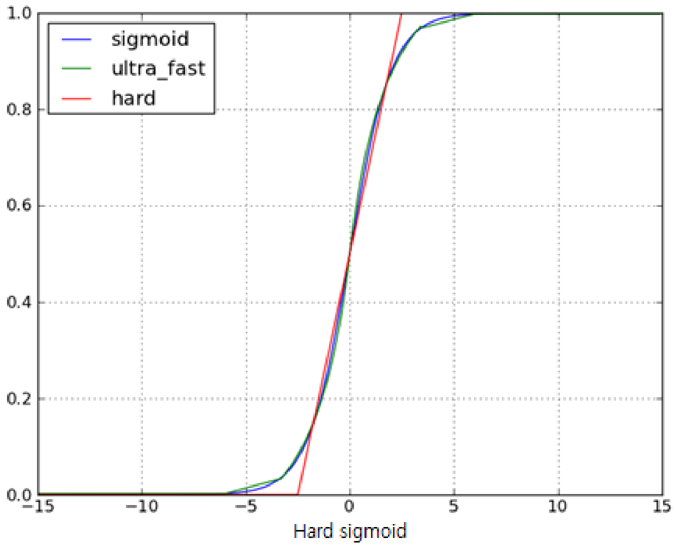
1	1	-1	-1
1	-1	-1	1
1	1	-1	1
-1	1	-1	1

Stochastic Binarization

$$x^b = \begin{cases} +1 & \text{with probability } p = \sigma(x), \\ -1 & \text{with probability } 1 - p, \end{cases} \qquad (2)$$

where

$$\sigma(x) = \mathrm{clip}(\frac{x+1}{2}, 0, 1) = \max(0, \min(1, \frac{x+1}{2})). \qquad (3)$$

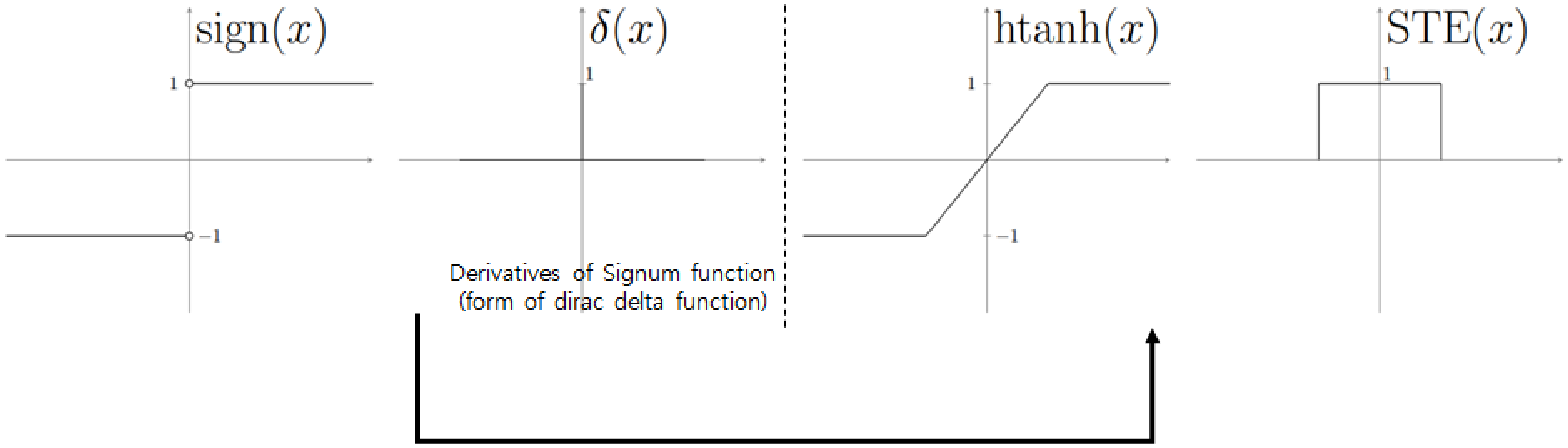


Hubara, Itay, et al. "Binarized neural networks." Advances in neural information processing systems. 2016.

기술적인 해결과제 : BNN 구현

Straight-through estimator?

an estimator g_q of the gradient $\frac{\partial C}{\partial q}$ $\longrightarrow$ straight-through estimator of $\frac{\partial C}{\partial r}$



$$g_r = g_q \underline{1_{|r|\leq 1}}.$$

$$\mathrm{Htanh}(x) = \mathrm{Clip}(x, -1, 1) = \max(-1, \min(1, x)). \quad (5)$$

Figure from Darabi, Sajad, et al. "BNN+: Improved binary network training." arXiv preprint arXiv:1812.11800 (2018).

기술적인 해결과제 : BNN 구현

Algorithm 1*

Algorithm 1 Training a BNN. C is the cost function for minibatch, λ - the learning rate decay factor and L the number of layers. $\circ$ indicates element-wise multiplication. The function Binarize() specifies how to (stochastically or deterministically) binarize the activations and weights, and Clip(), how to clip the weights. BatchNorm() specifies how to batch-normalize the activations, using either batch normalization (Ioffe & Szegedy, 2015) or its shift-based variant we describe in Algorithm 3. BackBatchNorm() specifies how to backpropagate through the normalization. Update() specifies how to update the parameters when their gradients are known, using either ADAM (Kingma & Ba, 2014) or the shift-based AdaMax we describe in Algorithm 4.

Require: a minibatch of inputs and targets (a_0, a^*), previous weights W, previous BatchNorm parameters θ, weights initialization coefficients from (Glorot & Bengio, 2010) γ, and previous learning rate η.

Ensure: updated weights W^{t+1}, updated BatchNorm parameters θ^{t+1} and updated learning rate η^{t+1}.

{1. Computing the parameters gradients:}
{1.1. Forward propagation:}
for $k = 1$ to L **do**
  $W_k^b \leftarrow \mathrm{Binarize}(W_k)$
  $s_k \leftarrow a_{k-1}^b W_k^b$
  $a_k \leftarrow \mathrm{BatchNorm}(s_k, \theta_k)$
  if $k < L$ **then**
    $a_k^b \leftarrow \mathrm{Binarize}(a_k)$
  end if
end for
{1.2. Backward propagation:}
{Please note that the gradients are not binary.}
Compute $g_{a_L} = \frac{\partial C}{\partial a_L}$ knowing a_L and a^*
for $k = L$ to 1 **do**
  if $k < L$ **then**
    $g_{a_k} \leftarrow g_{a_k^b} \circ 1_{|a_k| \leq 1}$
  end if
  $(g_{s_k}, g_{\theta_k}) \leftarrow \mathrm{BackBatchNorm}(g_{a_k}, s_k, \theta_k)$
  $g_{a_{k-1}^b} \leftarrow g_{s_k} W_k^b$
  $g_{W_k^b} \leftarrow g_{s_k}^\top a_{k-1}^b$
end for
{2. Accumulating the parameters gradients:}
for $k = 1$ to L **do**
  $\theta_k^{t+1} \leftarrow \mathrm{Update}(\theta_k, \eta, g_{\theta_k})$
  $W_k^{t+1} \leftarrow \mathrm{Clip}(\mathrm{Update}(W_k, \gamma_k \eta, g_{W_k^b}), -1, 1)$
  $\eta^{t+1} \leftarrow \lambda \eta$
end for

---

*Hubara, Itay, et al. "Binarized neural networks." Advances in neural information processing systems. 2016.

기술적인 해결과제 : BNN 구현

Algorithm 1: Training a BNN

Ensure: updated weights W^{t+1}, updated BatchNorm parameters θ^{t+1} and updated learning rate η^{t+1}.
{1. Computing the parameters gradients:}
{1.1. Forward propagation:}
for $k = 1$ to L **do**
  $W_k^b \leftarrow \text{Binarize}(W_k)$
  $s_k \leftarrow a_{k-1}^b W_k^b$
  $a_k \leftarrow \text{BatchNorm}(s_k, \theta_k)$
  if $k < L$ **then**
    $a_k^b \leftarrow \text{Binarize}(a_k)$
  end if
end for
{1.2. Backward propagation:}
{Please note that the gradients are not binary.}
Compute $g_{a_L} = \frac{\partial C}{\partial a_L}$ knowing a_L and a^*
for $k = L$ to 1 **do**
  if $k < L$ **then**
    $g_{a_k} \leftarrow g_{a_k^b} \circ 1_{|a_k| \leq 1}$
  end if
  $(g_{s_k}, g_{\theta_k}) \leftarrow \text{BackBatchNorm}(g_{a_k}, s_k, \theta_k)$
  $g_{a_{k-1}^b} \leftarrow g_{s_k} W_k^b$
  $g_{W_k^b} \leftarrow g_{s_k}^\top a_{k-1}^b$
end for
{2. Accumulating the parameters gradients:}
for $k = 1$ to L **do**
  $\theta_k^{t+1} \leftarrow \text{Update}(\theta_k, \eta, g_{\theta_k})$
  $W_k^{t+1} \leftarrow \text{Clip}(\text{Update}(W_k, \gamma_k \eta, g_{W_k^b}), -1, 1)$
  $\eta^{t+1} \leftarrow \lambda \eta$
end for

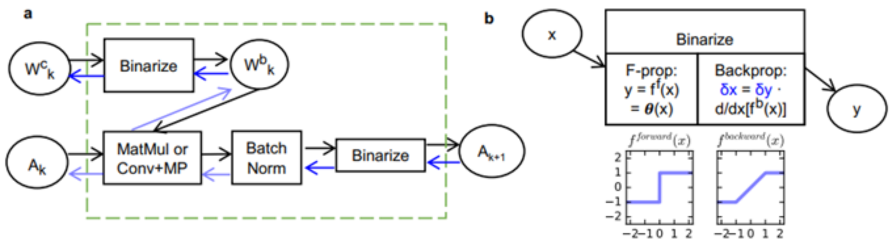
- Binary Activation : Signum Activation Layer (backward: using STE)
- Binary Weight : When using a weight w^r, quantize it using w^b=Sign(w^r)

Constrain each real-valued weight between -1 and 1.

→ Clip weights during training

The real-valued weights would otherwise grow very large

without any impact on the binary weights.



Anderson, Alexander G., and Cory P. Berg. "The high-dimensional geometry of binary neural networks." ICLR2018.

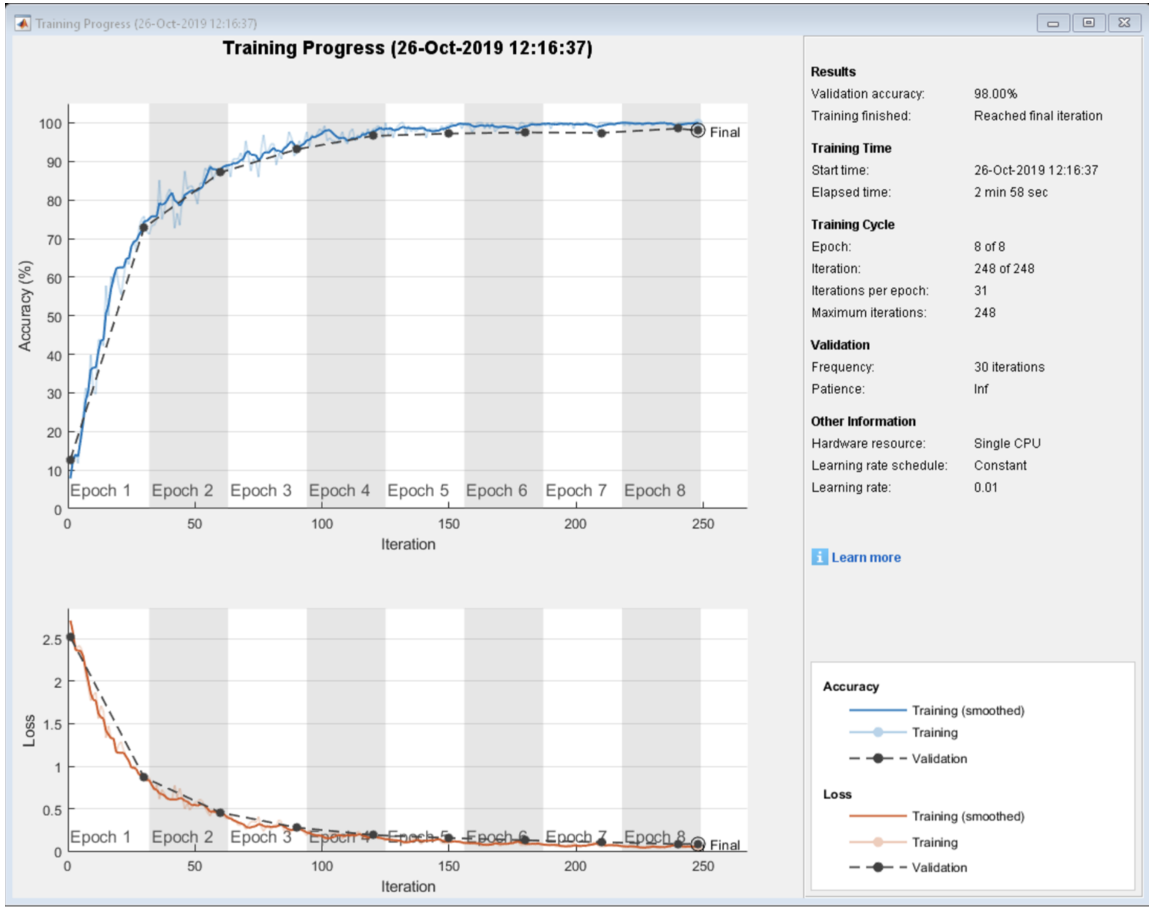
MathWorks 솔루션을 통한 해결 방안 및 결과

기존 함수를 활용한 Model Training

```
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
```

```
options = trainingOptions('sgdm', ...
    'MaxEpochs',8, ...
    'ValidationData',{XValidation,YValidation}, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

```
net = trainNetwork(XTrain,YTrain,layers,options);
```



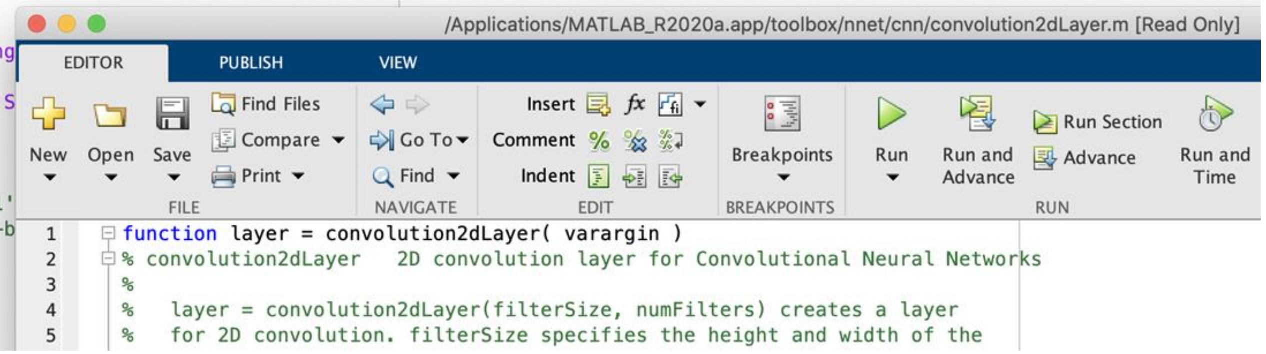
https://kr.mathworks.com/help/deeplearning/ref/trainingoptions.html

MathWorks 솔루션을 통한 해결 방안 및 결과

내부 구조는 어떻게 생겼나?

```
layers = [
    imageInputLayer([imgsize 3],'Name','Input','AverageImage', img_mean)  %% Should have Mean

    convolution2dLayer(5,64,'Padding',2,'Stride',1,'Name','Conv1')
    batchNormalizationLayer('Name','bn1')
    reluLayer('Name','r1')
    maxPooling2dLayer(3,'Stride',2,'Padding',1,'Name','Pool1')

    convolution2dLayer(5,64,'Padding',2,'Stride',1,'Name','Conv2')
    batchNormalizationLayer('Name','bn2')
    reluLayer('Name','r2')
    maxPooling2dLayer(3,'Stride',2,'Padding

    convolution2dLayer(5,128,'Padding',2,'S
    batchNormalizationLayer('Name','bn3')
    reluLayer('Name','r3')

%     fullyConnectedLayer(3072,'Name','fc1'
%     batchNormalizationLayer('Name','fc1-b
%     reluLayer('Name','fc1-r')
%     dropoutLayer(0.5,'Name','drop1')
    ];
```

ctrl+D (Mac은 command+shift+D)를 통해 라이브러리 함수 열람



https://kr.mathworks.com/help/deeplearning/ref/trainingoptions.html

MathWorks 솔루션을 통한 해결 방안 및 결과

Convolution Layers를 살펴보면!

convolution2dLayer -> Convolution2D -> **Executionstrategy**
-> Convolution2DLayer

```
function layer = convolution2dLayer( varargin )
% convolution2dLayer   2D convolution layer for Convolutional Neural Networ

    internalLayer = nnet.internal.cnn.layer.Convolution2D(args.Name, ...
                                                          args.FilterSize, ...
                                                          args.NumChannels, ...
                                                          args.NumFilters, ...
                                                          args.Stride, ...
                                                          args.DilationFactor,
                                                          args.PaddingMode, ...
                                                          args.PaddingSize);

    layer = nnet.cnn.layer.Convolution2DLayer(internalLayer);
    layer.WeightsInitializer = args.WeightsInitializer;
    layer.BiasInitializer = args.BiasInitializer;
    layer.Weights = args.Weights;
    layer.Bias = args.Bias;
end
```

```
classdef Convolution2D < nnet.internal.cnn.layer.FunctionalLayer ...
        & nnet.internal.cnn.layer.CPUFusableLayer
    % Convolution2D   Implementation of the 2D convolution layer

    %   Copyright 2015-2019 The MathWorks, Inc.

        function this = Convolution2D( ...
                name, filterSize, numChannels, numFilters, stride, dilationFactor, paddingMode, paddingSize)

        function Z = predict( this, X )
            % predict   Forward input data through the layer and output the
            if(this.usingFilterGroups())
                Z = this.predictTwoFilterGroupsWithCaching( X );
            else
                Z = this.predictNormal( X );
            end
        end

        function [Z, memory] = forward( this, X )
            % forward   Forward propagate data during training
            memory = [];
            if(this.usingFilterGroups())
                Z = this.forwardTwoFilterGroups( X );
            else
                Z = this.forwardNormal( X );
            end
        end

        function varargout = backward( this, X, ~, dZ, ~ )
            % backward    Back propagate the derivative of the loss functio
            % through the layer
            if(this.usingFilterGroups())
                [varargout{1:nargout}] = this.backwardTwoFilterGroups(X, [],
            else
                [varargout{1:nargout}] = this.backwardNormal(X, [], dZ, []);
            end
        end
```

```
        function this = setHostStrategy(this)
            % setHostStrategy   Use Mkldnn only if Mkldnn is featured on and no dilation
            noDilation = isequal(this.DilationFactor, [1 1]);
            if nnet.internal.cnnhost.useMKLDNN && noDilation
                this.ExecutionStrategy = nnet.internal.cnn.layer.util.Convolution2DHostMkldnnStrategy();
            else
                this.ExecutionStrategy = nnet.internal.cnn.layer.util.Convolution2DHostStridedConvStrategy();
            end
        end

        function this = setGPUStrategy(this)
            this.ExecutionStrategy = nnet.internal.cnn.layer.util.Convolution2DGPUStrategy();
        end
    end

    methods(Access=protected)
        function this = setFunctionalStrategy(this)
            this.ExecutionStrategy = ...
                nnet.internal.cnn.layer.util.Convolution2DFunctionalStrategy();
        end
    end
```

MathWorks 솔루션을 통한 해결 방안 및 결과

Convolution Layers를 살펴보면!

convolution2dLayer -> Convolution2D -> **Executionstrategy**
-> Convolution2DLayer

```matlab
function layer = convolution2dLayer( varargin )
% convolution2dLayer   2D convolution layer for Convolutional Neural Networ

    internalLayer = nnet.internal.cnn.layer.Convolution2D(args.Name, ...
                                                   args.FilterSize, ...
                                                   args.NumChannels, ...
                                                   args.NumFilters, ...
                                                   args.Stride, ...
                                                   args.DilationFactor,
                                                   args.PaddingMode, ...
                                                   args.PaddingSize);

    layer = nnet.cnn.layer.Convolution2DLayer(internalLayer);
    layer.WeightsInitializer = args.WeightsInitializer;
    layer.BiasInitializer = args.BiasInitializer;
    layer.Weights = args.Weights;
    layer.Bias = args.Bias;
end
```

```matlab
classdef Convolution2DLayer < nnet.cnn.layer.Layer & ...
        nnet.internal.cnn.layer.Externalizable
    % Convolution2DLayer   2-D convolution layer
    %
    %   To create a convolution layer, use convolution2dLayer
```

```matlab
classdef Convolution2D < nnet.internal.cnn.layer.FunctionalLayer ...
        & nnet.internal.cnn.layer.CPUFusableLayer
    % Convolution2D   Implementation of the 2D convolution layer

    %   Copyright 2015-2019 The MathWorks, Inc.

        function this = Convolution2D( ...
                name, filterSize, numChannels, numFilters, stride, dilationFactor, paddingMode, paddingSize)

        function Z = predict( this, X )
            % predict   Forward input data through the layer and output the
            if(this.usingFilterGroups())
                Z = this.predictTwoFilterGroupsWithCaching( X );
            else
                Z = this.predictNormal( X );
            end
        end

        function [Z, memory] = forward( this, X )
            % forward   Forward propagate data during training
            memory = [];
            if(this.usingFilterGroups())
                Z = this.forwardTwoFilterGroups( X );
            else
                Z = this.forwardNormal( X );
            end
        end

        function varargout = backward( this, X, ~, dZ, ~ )
            % backward    Back propagate the derivative of the loss functio
            % through the layer
            if(this.usingFilterGroups())
                [varargout{1:nargout}] = this.backwardTwoFilterGroups(X, [],
            else
                [varargout{1:nargout}] = this.backwardNormal(X, [], dZ, []);
            end
        end
```

```matlab
        function this = setHostStrategy(this)
            % setHostStrategy   Use Mkldnn only if Mkldnn is featured on and no dilation
            noDilation = isequal(this.DilationFactor, [1 1]);
            if nnet.internal.cnnhost.useMKLDNN && noDilation
                this.ExecutionStrategy = nnet.internal.cnn.layer.util.Convolution2DHostMkldnnStrategy();
            else
                this.ExecutionStrategy = nnet.internal.cnn.layer.util.Convolution2DHostStridedConvStrategy();
            end
        end

        function this = setGPUStrategy(this)
            this.ExecutionStrategy = nnet.internal.cnn.layer.util.Convolution2DGPUStrategy();
        end
    end

    methods(Access=protected)
        function this = setFunctionalStrategy(this)
            this.ExecutionStrategy = ...
                nnet.internal.cnn.layer.util.Convolution2DFunctionalStrategy();
        end
    end
```

MathWorks 솔루션을 통한 해결 방안 및 결과

Convolution Layers를 살펴보면!

Execution strategy

```matlab
classdef Convolution2DGPUStrategy < nnet.internal.cnn.layer.util.ExecutionSt
    % Convolution2DGPUStrategy   Execution strategy for running the
    % convolution on the GPU

    methods
        function [Z, memory] = forward(~, X, ...
                weights, bias, ...
                topPad, leftPad, ...
                bottomPad, rightPad, ...
                verticalStride, horizontalStride, ...
                verticalDilation, horizontalDilation)
            paddingSize = [topPad bottomPad leftPad rightPad];
            if iPaddingIsSymmetric(paddingSize)
                Z = nnet.internal.cnngpu.convolveForward2D( ...
                    X, weights, ...
                    topPad, leftPad, ...
                    bottomPad, rightPad, ...
                    verticalStride, horizontalStride, ...
                    verticalDilation, horizontalDilation) + bias;
            else
                X = iPadArray(X, paddingSize);
                Z = nnet.internal.cnngpu.convolveForward2D( ...
                    X, weights, ...
                    0, 0, ...
                    0, 0, ...
                    verticalStride, horizontalStride, ...
                    verticalDilation, horizontalDilation) + bias;
            end
            memory = [];
        end
```

```matlab
        function [dX,dW] = backward(~, ...
                X, weights, dZ, ...
                topPad, leftPad, ...
                bottomPad, rightPad, ...
                strideHeight, strideWidth, ...
                verticalDilation, horizontalDilation)
            paddingSize = [topPad bottomPad leftPad rightPad];
            needsWeightGradients = nargout > 1;
            if iPaddingIsSymmetric(paddingSize)
                dX = nnet.internal.cnngpu.convolveBackwardData2D( ...
                    X, weights, dZ, ...
                    topPad, leftPad, ...
                    bottomPad, rightPad, ...
                    strideHeight, strideWidth, ...
                    verticalDilation, horizontalDilation);
                if needsWeightGradients
                    dW{1} = nnet.internal.cnngpu.convolveBackwardFilter2D( .
                        X, weights, dZ, ...
                        topPad, leftPad, ...
                        bottomPad, rightPad, ...
                        strideHeight, strideWidth, ...
                        verticalDilation, horizontalDilation);
                end
            else
                X = iPadArray(X, paddingSize);
                dX = nnet.internal.cnngpu.convolveBackwardData2D( ...
                    X, weights, dZ, ...
                    0, 0, ...
                    0, 0, ...
                    strideHeight, strideWidth, ...
                    verticalDilation, horizontalDilation);
                dX = iUnpadArray(dX, paddingSize);
```

MathWorks 솔루션을 통한 해결 방안 및 결과

Binarized Convolutional Layers

Ensure: updated weights W^{t+1}, updated BatchNorm parameters θ^{t+1} and updated learning rate η^{t+1}.

{1. Computing the parameters gradients:}

{1.1. Forward propagation:}

for $k = 1$ to L **do**

  $W_k^b \leftarrow \text{Binarize}(W_k)$

  $s_k \leftarrow a_{k-1}^b W_k^b$

  $a_k \leftarrow \text{BatchNorm}(s_k, \theta_k)$

  if $k < L$ **then**

    $a_k^b \leftarrow \text{Binarize}(a_k)$

  end if

end for

{1.2. Backward propagation:}

{Please note that the gradients are not binary.}

Compute $g_{a_L} = \frac{\partial C}{\partial a_L}$ knowing a_L and a^*

for $k = L$ to 1 **do**

  if $k < L$ **then**

    $g_{a_k} \leftarrow g_{a_k^b} \circ 1_{|a_k| \le 1}$

  end if

  $(g_{s_k}, g_{\theta_k}) \leftarrow \text{BackBatchNorm}(g_{a_k}, s_k, \theta_k)$

  $g_{a_{k-1}^b} \leftarrow g_{s_k} W_k^b$

  $g_{W_k^b} \leftarrow g_{s_k}^\top a_{k-1}^b$

end for

{2. Accumulating the parameters gradients:}

for $k = 1$ to L **do**

  $\theta_k^{t+1} \leftarrow \text{Update}(\theta_k, \eta, g_{\theta_k})$

  $W_k^{t+1} \leftarrow \text{Clip}(\text{Update}(W_k, \gamma_k \eta, g_{W_k^b}), -1, 1)$

  $\eta^{t+1} \leftarrow \lambda \eta$

end for

다른 이름으로 저장 후 수정!

```
        function [Z, memory] = forward(~, X, ...
                weights, bias, ...
                topPad, leftPad, ...
                bottomPad, rightPad, ...
                verticalStride, horizontalStride, ...
                verticalDilation, horizontalDilation)
            paddingSize = [topPad bottomPad leftPad rightPad];

            weights= sign(weights);
            weights(weights==0)=1;
```

```
        function [dX,dW] = backward(~, ...
                X, weights, dZ, ...
                topPad, leftPad, ...
                bottomPad, rightPad, ...
                strideHeight, strideWidth, ...
                verticalDilation, horizontalDilation)
            paddingSize = [topPad bottomPad leftPad rightPad];
            needsWeightGradients = nargout > 1;

            weightsFP=weights;
            weights= sign(weights);
            weights(weights==0)=1;
```

이진화 컨벌루션 함수 생성!

- Binarizedconvolution2dLayer.m
- BinarizedConvolution2D.m
- BinarizedConvolution2DFunctionalStrategy.m
- BinarizedConvolution2DGPUStrategy.asv
- BinarizedConvolution2DGPUStrategy.m
- BinarizedConvolution2DHostMkldnnStrategy.m
- BinarizedConvolution2DHostStridedConvStrategy.m

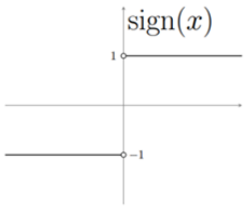
MathWorks 솔루션을 통한 해결 방안 및 결과

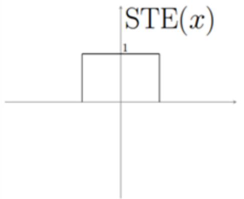
Custom Layer를 활용하여 필요한 활성 함수 만들기!

```
classdef myLayer < nnet.layer.Layer

    properties
        % (Optional) Layer properties.

        % Layer properties go here.
    end

    properties (Learnable)
        % (Optional) Layer learnable parameters.

        % Layer learnable parameters go here.
    end

    methods
        function layer = myLayer()
            % (Optional) Create a myLayer.
            % This function must have the same name as the class.

            % Layer constructor function goes here.
        end

        function [Z1, …, Zm] = predict(layer, X1, …, Xn)
            % Forward input data through the layer at prediction time and
            % output the result.
            %
            % Inputs:
            %         layer       - Layer to forward propagate through
            %         X1, ..., Xn - Input data
            % Outputs:
            %         Z1, ..., Zm - Outputs of layer forward function

            % Layer forward function for prediction goes here.
        end
```

Sign Activation Layer → Binary Activation!

```
classdef SignumActivation < nnet.layer.Layer
    methods
        function layer = SignumActivation(name)
            layer.Name = name;
        end

        function [Z,memory] = foward(~, X)
            Z=sign(X);
            Z(Z==0)=1;
            memory=[];
        end

        function Z = predict(~, X)
            Z=sign(X);
            Z(Z==0)=1;
        end

        function dLdX = backward(~, X, ~, dLdZ, ~)
            dLdX = (dLdZ) .*(X>-1 & X<1) ;
        end
    end
end
```





SignumActivation.m

https://kr.mathworks.com/help/deeplearning/ug/define-custom-deep-learning-layers.html

MathWorks 솔루션을 통한 해결 방안 및 결과

New Functions for BNN Training

Ensure: updated weights W^{t+1}, updated BatchNorm parameters θ^{t+1} and updated learning rate η^{t+1}.

{1. Computing the parameters gradients:}

{1.1. Forward propagation:}

for $k = 1$ to L **do**
- $W_k^b \leftarrow \text{Binarize}(W_k)$
- $s_k \leftarrow a_{k-1}^b W_k^b$
- $a_k \leftarrow \text{BatchNorm}(s_k, \theta_k)$
- **if** $k < L$ **then**
  - $a_k^b \leftarrow \text{Binarize}(a_k)$
- **end if**

end for

{1.2. Backward propagation:}

{Please note that the gradients are not binary.}

Compute $g_{a_L} = \frac{\partial C}{\partial a_L}$ knowing a_L and a^*

for $k = L$ to 1 **do**
- **if** $k < L$ **then**
  - $g_{a_k} \leftarrow g_{a_k^b} \circ 1_{|a_k| \leq 1}$
- **end if**
- $(g_{s_k}, g_{\theta_k}) \leftarrow \text{BackBatchNorm}(g_{a_k}, s_k, \theta_k)$
- $g_{a_{k-1}^b} \leftarrow g_{s_k} W_k^b$
- $g_{W_k^b} \leftarrow g_{s_k}^\top a_{k-1}^b$

end for

{2. Accumulating the parameters gradients:}

for $k = 1$ to L **do**
- $\theta_k^{t+1} \leftarrow \text{Update}(\theta_k, \eta, g_{\theta_k})$
- $W_k^{t+1} \leftarrow \text{Clip}(\text{Update}(W_k, \gamma_k \eta, g_{W_k^b}), -1, 1)$
- $\eta^{t+1} \leftarrow \lambda \eta$

end for

CNN의 1-bit quantization을 위한 함수들

- Binarizedconvolution2dLayer.m
- BinarizedConvolution2D.m
- BinarizedConvolution2DFunctionalStrategy.m
- BinarizedConvolution2DGPUStrategy.asv
- BinarizedConvolution2DGPUStrategy.m
- BinarizedConvolution2DHostMkldnnStrategy.m
- BinarizedConvolution2DHostStridedConvStrategy.m
- SignumActivation.m

```
layers = [
    imageInputLayer([32 32 3],'Name','input','Normalization','none')

    Binarizedconvolution2dLayer(3,64,'Padding','same','Stride',1,'BiasLearnRateFactor',0,'Name','binConv1')
    batchNormalizationLayer('Name','BatchNorm1')
    SignumActivation('Sign1')
    Binarizedconvolution2dLayer(3,64,'Padding','same','Stride',1,'BiasLearnRateFactor',0,'Name','binConv2')
    batchNormalizationLayer('Name','BatchNorm2')
    SignumActivation('Sign2')

    maxPooling2dLayer(2,'Stride',2,'Name','MaxPool1')

    Binarizedconvolution2dLayer(3,128,'Padding','same','Stride',1,'BiasLearnRateFactor',0,'Name','binConv3')
    batchNormalizationLayer('Name','BatchNorm3')
    SignumActivation('Sign3')
    Binarizedconvolution2dLayer(3,128,'Padding','same','Stride',1,'BiasLearnRateFactor',0,'Name','binConv4')
    batchNormalizationLayer('Name','BatchNorm4')
    SignumActivation('Sign4')

    maxPooling2dLayer(2,'Stride',2,'Name','MaxPool2')

    Binarizedconvolution2dLayer(3,256,'Padding','same','Stride',1,'BiasLearnRateFactor',0,'Name','binConv5')
    batchNormalizationLayer('Name','BatchNorm5')
    SignumActivation('Sign5')
    Binarizedconvolution2dLayer(3,256,'Padding','same','Stride',1,'BiasLearnRateFactor',0,'Name','binConv6')
    batchNormalizationLayer('Name','BatchNorm6')
    SignumActivation('Sign6')

    maxPooling2dLayer(2,'Stride',2,'Name','MaxPool3')

    averagePooling2dLayer(4,'Name','avePool1')
    SignumActivation('SignAve')
    %======= :Classifier: =======%
    Binarizedconvolution2dLayer(1,10,'Padding','same','Stride',1,'BiasLearnRateFactor',0,'Name','binConv7')
    batchNormalizationLayer('Name','BatchNorm7')
    softmaxLayer('Name','softmax')
    classificationLayer('Name','classoutput')
    ];
```

BNN(VGG7 model)

MathWorks 솔루션을 통한 해결 방안 및 결과

CIFAR-10 Training Results

Validation Data Confusion Matrix

Output Class \ Target Class	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	
airplane	865 8.6%	9 0.1%	45 0.4%	23 0.2%	22 0.2%	13 0.1%	2 0.0%	13 0.1%	62 0.6%	20 0.2%	80.5% 19.5%
automobile	16 0.2%	931 9.3%	1 0.0%	8 0.1%	2 0.0%	2 0.0%	1 0.0%	3 0.0%	15 0.1%	45 0.4%	90.9% 9.1%
bird	34 0.3%	2 0.0%	750 7.5%	41 0.4%	42 0.4%	25 0.3%	31 0.3%	12 0.1%	9 0.1%	4 0.0%	78.9% 21.1%
cat	14 0.1%	1 0.0%	25 0.3%	630 6.3%	25 0.3%	99 1.0%	26 0.3%	14 0.1%	1 0.0%	4 0.0%	75.1% 24.9%
deer	9 0.1%	0 0.0%	44 0.4%	56 0.6%	805 8.1%	30 0.3%	12 0.1%	32 0.3%	4 0.0%	3 0.0%	80.9% 19.1%
dog	4 0.0%	0 0.0%	39 0.4%	122 1.2%	18 0.2%	750 7.5%	8 0.1%	25 0.3%	0 0.0%	1 0.0%	77.6% 22.4%
frog	5 0.1%	1 0.0%	56 0.6%	69 0.7%	49 0.5%	35 0.4%	909 9.1%	8 0.1%	7 0.1%	2 0.0%	79.7% 20.3%
horse	8 0.1%	2 0.0%	27 0.3%	25 0.3%	32 0.3%	35 0.4%	4 0.0%	885 8.8%	2 0.0%	2 0.0%	86.6% 13.4%
ship	33 0.3%	9 0.1%	5 0.1%	8 0.1%	3 0.0%	5 0.1%	4 0.0%	1 0.0%	884 8.8%	11 0.1%	91.8% 8.2%
truck	12 0.1%	45 0.4%	8 0.1%	18 0.2%	2 0.0%	6 0.1%	3 0.0%	7 0.1%	16 0.2%	908 9.1%	88.6% 11.4%
	86.5% 13.5%	93.1% 6.9%	75.0% 25.0%	63.0% 37.0%	80.5% 19.5%	75.0% 25.0%	90.9% 9.1%	88.5% 11.5%	88.4% 11.6%	90.8% 9.2%	83.2% 16.8%

Output Class

Target Class

TABLE III. COMPARISON OF BINARIZED VGG-11 CNNS (NOTE THAT, THE INTEGER CONVOLUTIONAL LAYER (ICONV1) USES 1 BIT WEIGHT AND 8 BIT INPUT.).

Layer	Baseline				Neuron Pruning				Proposed			
	Output Dim.	Input # Fmaps	Output Fmaps	Weight [bits]	Output Dim.	Input # Fmaps	Output Fmaps	Weight [bits]	Output Dim.	Input # Fmaps	Output Fmaps	Weight [bits]
IConv1	32× 32	3	64	1.7K	32× 32	3	64	1.7K	32× 32	3	64	1.7K
BConv2	32× 32	64	64	36.8K	32× 32	64	64	36.8K	32× 32	64	64	36.8K
Max Pool	16× 16	64	64		16× 16	64	64		16× 16	64	64	
BConv3	16× 16	64	128	73.7K	16× 16	64	128	73.7K	16× 16	64	128	73.7K
BConv4	16× 16	128	128	147.4K	16× 16	128	128	147.4K	16× 16	128	128	147.4K
Max Pool	8× 8	128	128		8× 8	128	128		8× 8	128	128	
BConv5	8× 8	128	256	294.9K	8× 8	128	256	294.9K	8× 8	128	256	294.9K
BConv6	8× 8	256	256	589.8K	8× 8	256	256	589.8K	8× 8	256	256	589.8K
Max Pool	4× 4	256	256		4× 4	256	256		4× 4	256	256	
BFC1	1× 1	4096	4096	16.7M	32× 32	4096	354	14.4M		(Ave Pool)		
BFC2	1× 1	4096	4096	16.7M	32× 32	354	31	10.9K				
BFC3	1× 1	4096	10	40.9K	32× 32	31	10	310	1× 1	256	10	2.5K
(fc total)				(33.6M)				(14.6M)				(2.5K)
Total				34.7M				26.0M				**11.5M**
Error Rate				18.6%				19.1%				18.2%

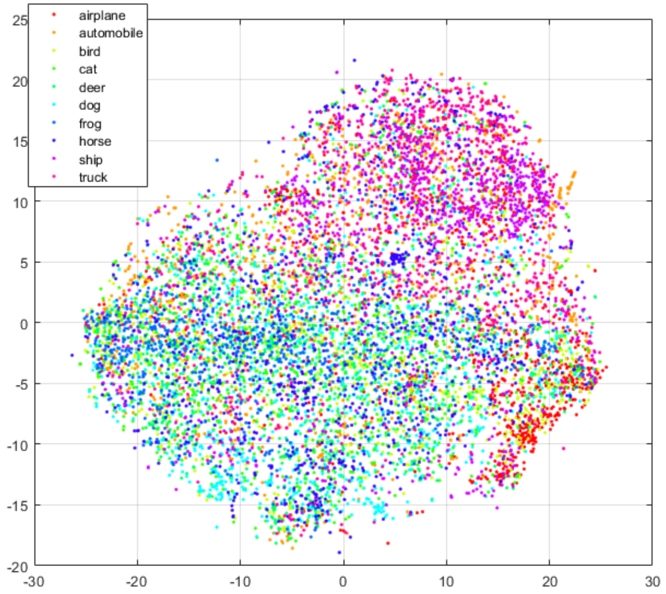
Error Rate = 18.2%

Paper Reference*

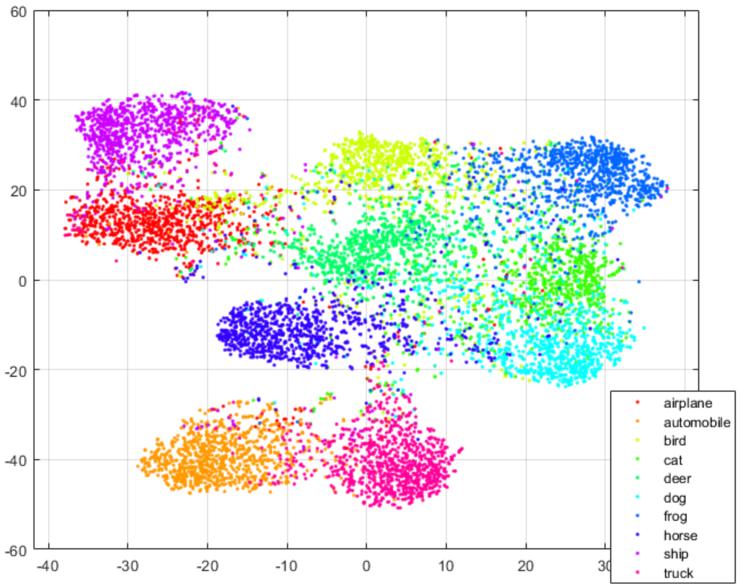
*Nakahara, Hiroki, Tomoya Fujii, and Shimpei Sato. "A fully connected layer elimination for a binarizec convolutional neural network on an FPGA." 2017 27th International Conference on Field Programmable Logic and Applications (FPL). IEEE, 2017.

MathWorks 솔루션을 통한 해결 방안 및 결과

Latent Space Visualization(t-SNE)



Raw

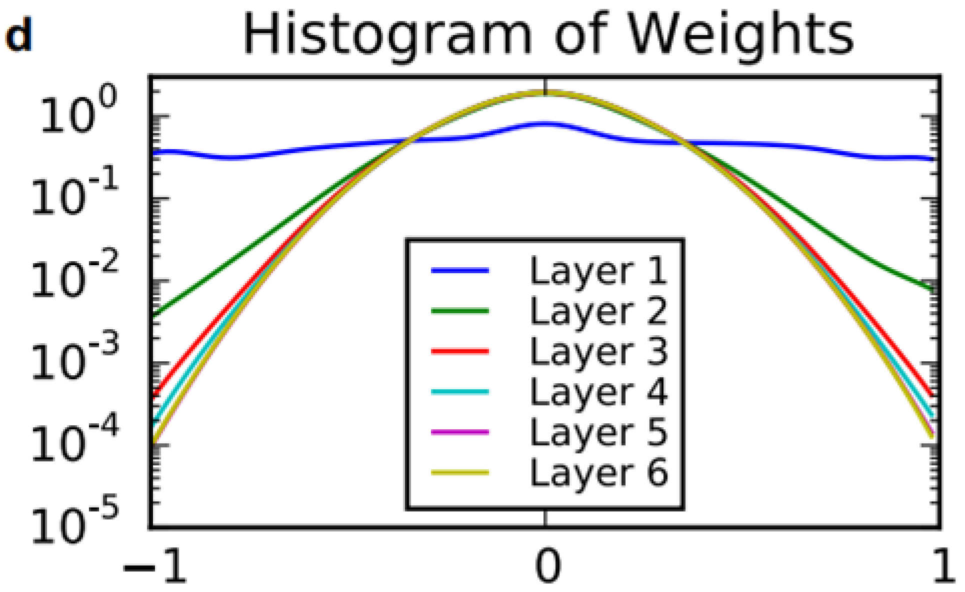


BNN Features

Nakahara, Hiroki, Tomoya Fujii, and Shimpei Sato. "A fully connected layer elimination for a binarizec convolutional neural network on an FPGA." 2017 27th International Conference on Field Programmable Logic and Applications (FPL). IEEE, 2017.

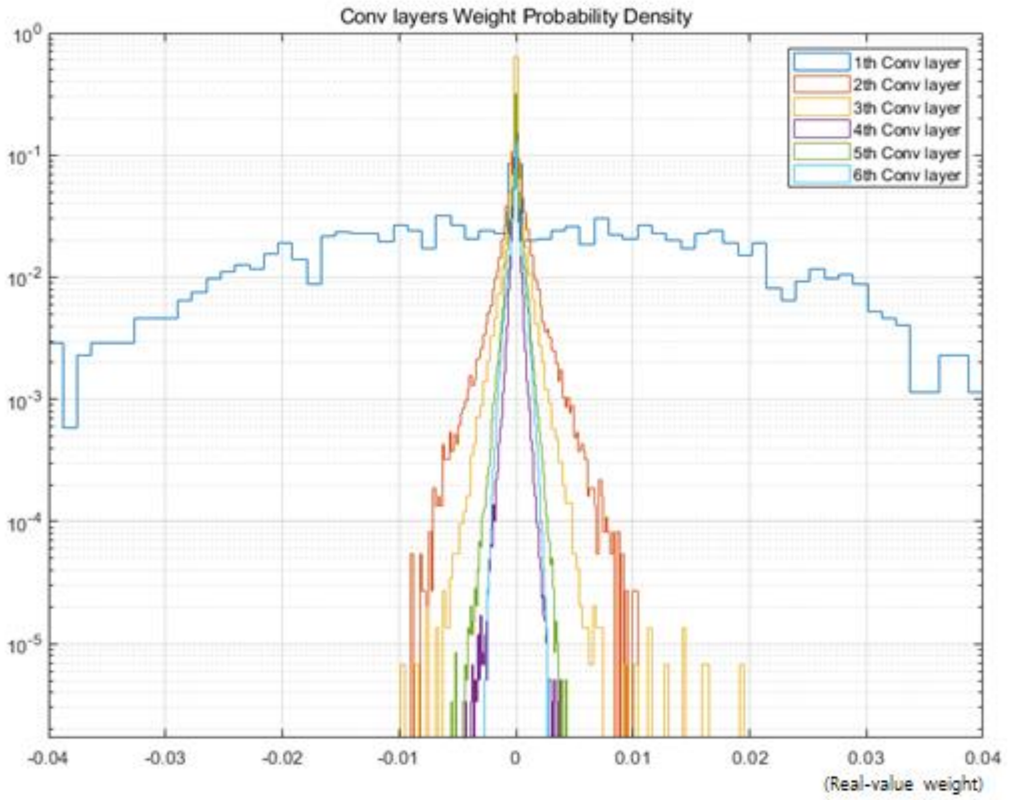
MathWorks 솔루션을 통한 해결 방안 및 결과

BNN Training Results



Anderson, Alexander G., and Cory P. Berg. "The high-dimensional geometry of binary neural networks." ICLR2018.

What paper said



What I've got

결과 및 정리 (Achievements and Outlook)

- BNN은 신경망 양자화(Quantization)기법 중 하나로 모델의 가중치와 활성화 값을 1-bit으로 양자화
- 1-bit 양자화를 통해 얻을 수 있는 이득으로는..
  1. 모델 사이즈 감소
  2. 메모리 소비 감소
  3. 연산 가속 효과 기대
  4. 하드웨어 구성 시 집적도 향상
- MATLAB을 통한 BNN 논문 구현 방법을 소개하고 결과물을 기존 MATLAB 함수들을 사용하여 평가

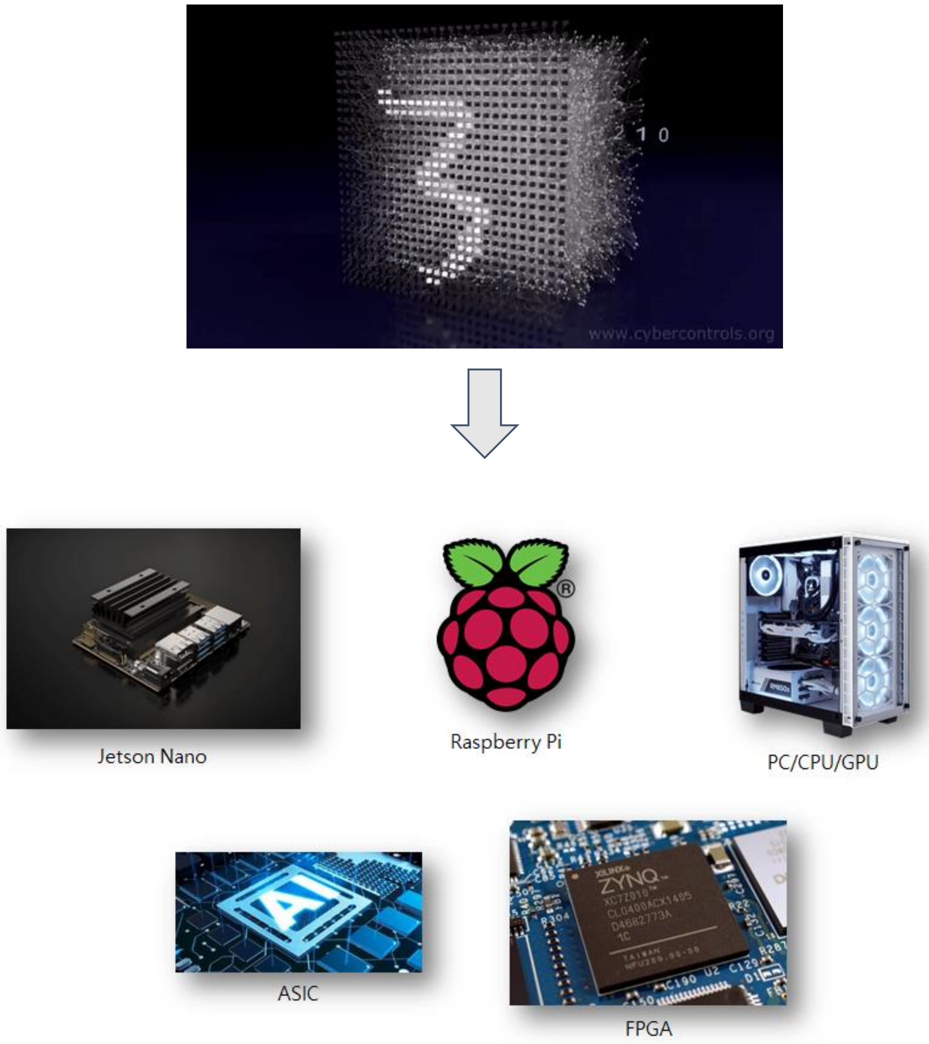
MATLAB으로 논문을 구현하며 좋았던 장점은…

- 레이어 별 함수 내부로 접근이 가능하여 구조를 살펴볼 수 있음
- 필요한 기능을 구현하기 위해 해당 함수 및 클래스에 직접 찾아가 모듈별로 직관적으로 구현 용이

MATLAB으로 구현하며 겪은 단점은..

- 모든 함수에 접근이 가능한 것은 아님
- 주석이나 설명이 불친절 한 경우나 함수가 복잡하게 얽혀 있는 경우가 많음

결론 (Concluding Remarks)



- 딥러닝 네트워크 모델은 최종 단계에서 다양한 환경에 배포됨
- 엣지 환경에서 클라우드 환경까지 다양한 환경에는 다양한 제약 조건이 존재
- 모델 배포 단계에서 타겟 환경을 고려한 경량화 및 압축 기술은 필수적
- 이를 위해 신경망의 양자화 및 모델 경량화 기술이 중요
- 오늘은 이러한 신경망의 양자화 중 1-bit(이진화) 신경망에 대하여 소개
- 양자화된 네트워크 모델은 추론 시에 가속기와 함께 할 때 효과가 극대화
- 각 환경에는 그 환경에 맞는 다양한 형태의 가속, 연산기가 존재
- 신경망 양자화는 모델 사이즈의 감소와 더불어 연산기와 함께 할 때 연산 가속, 하드웨어의 집적도, 메모리 소모 감소 등 다양한 이득을 얻을 수 있음
- Google Coral, Xilinx FPGA, Arm의 Ethos, Mali, Arm nn, NVIDIA의 TensorRT 등