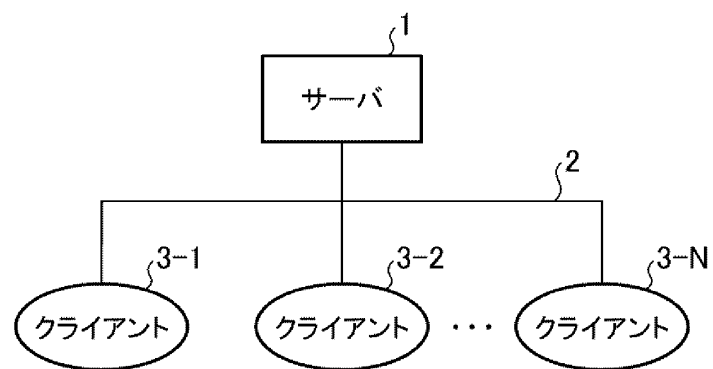
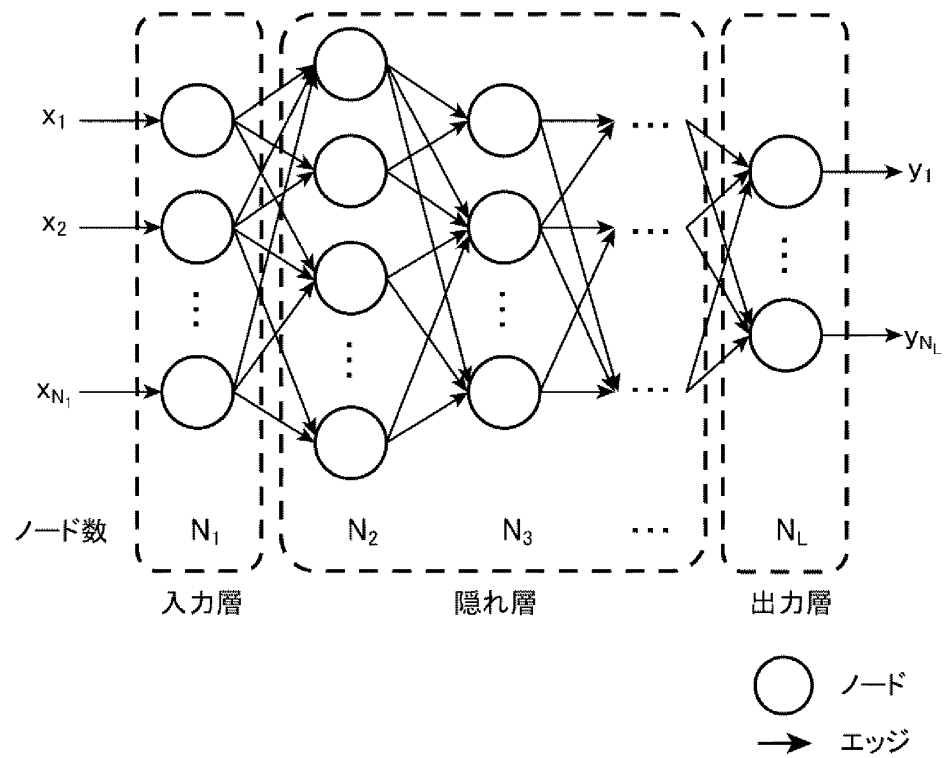


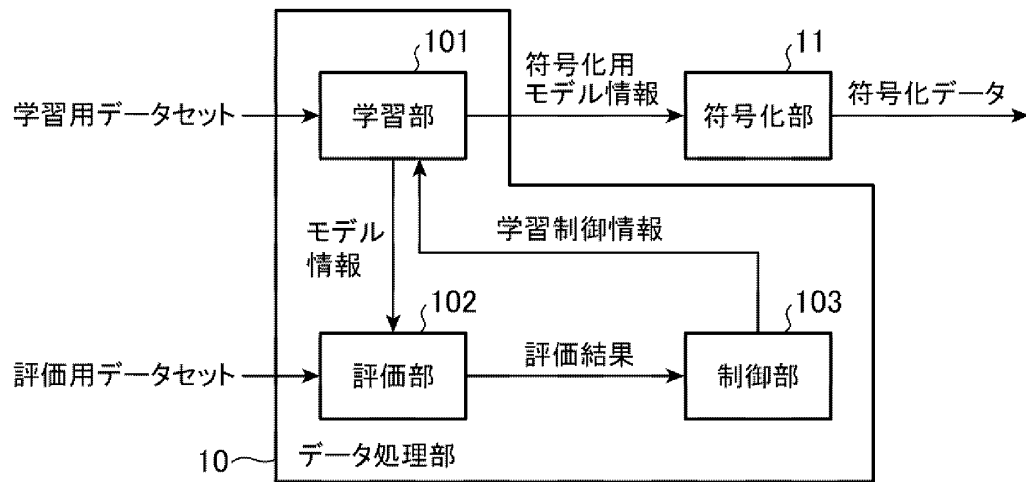
[図1]



[図2]



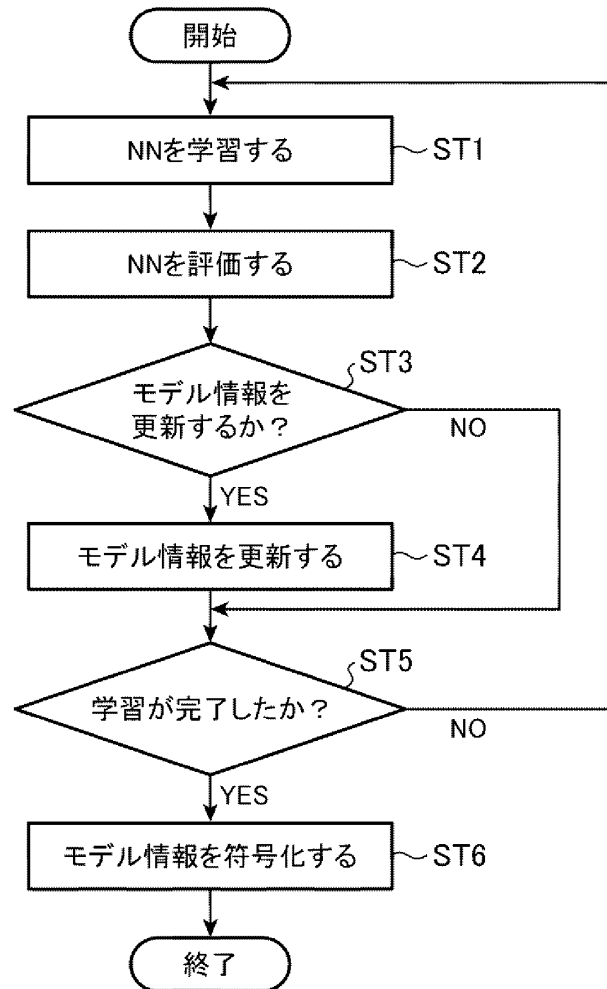
[図3]



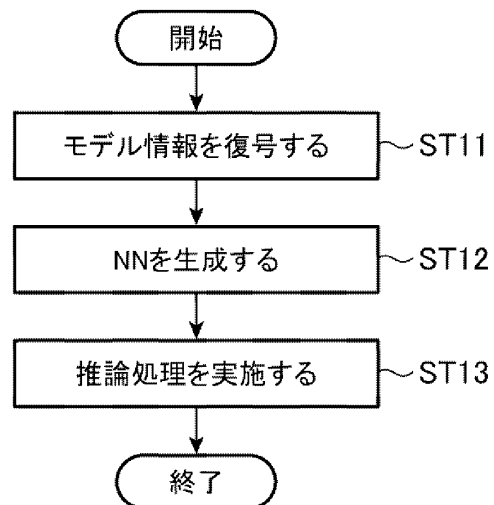
[図4]



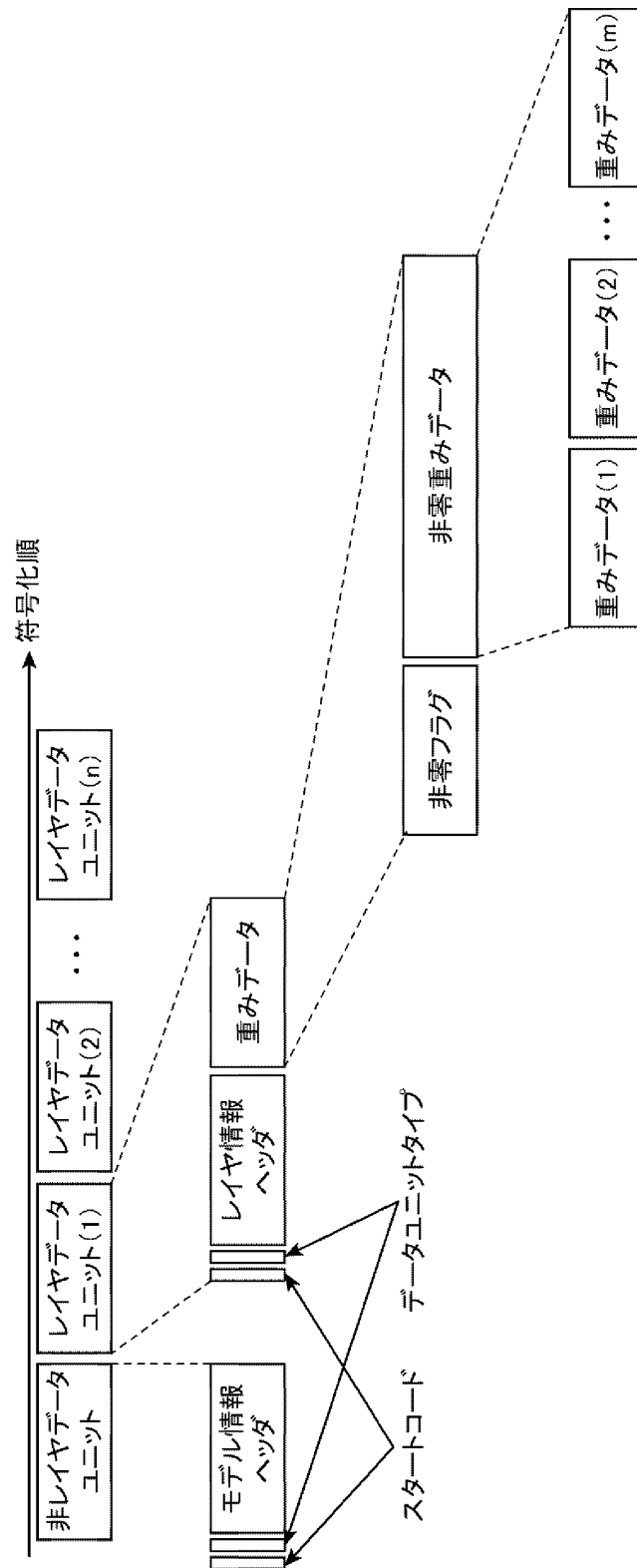
[図5]



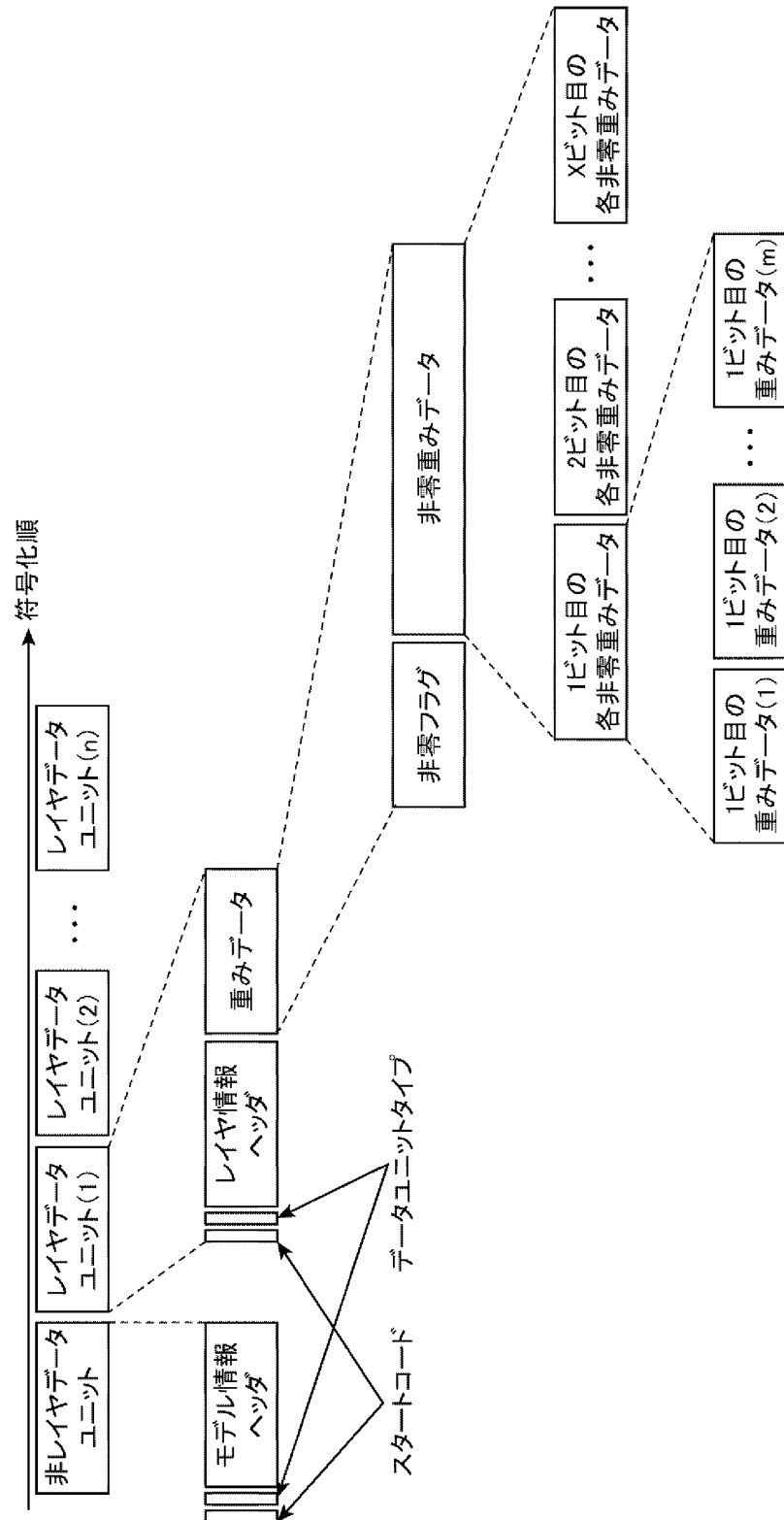
[図6]



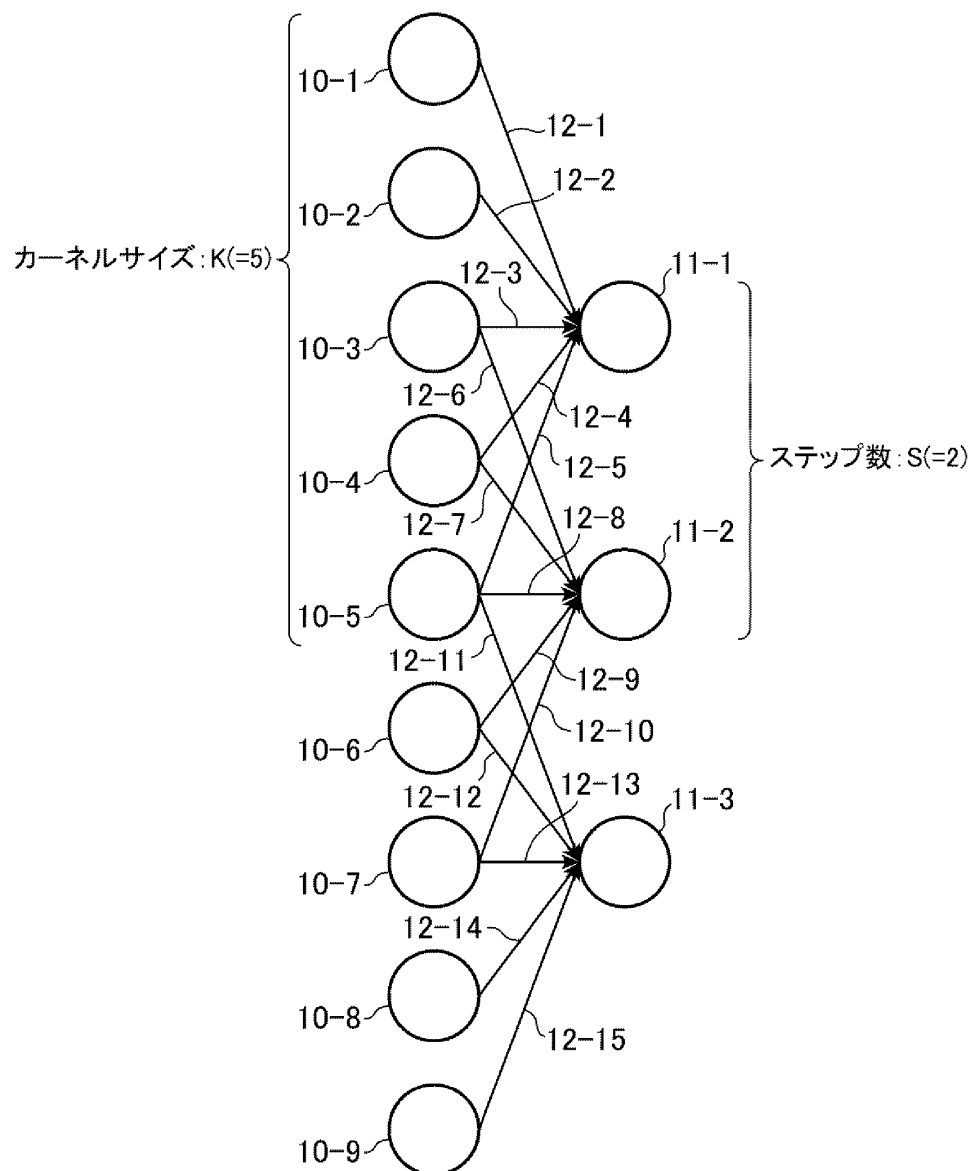
[図7]



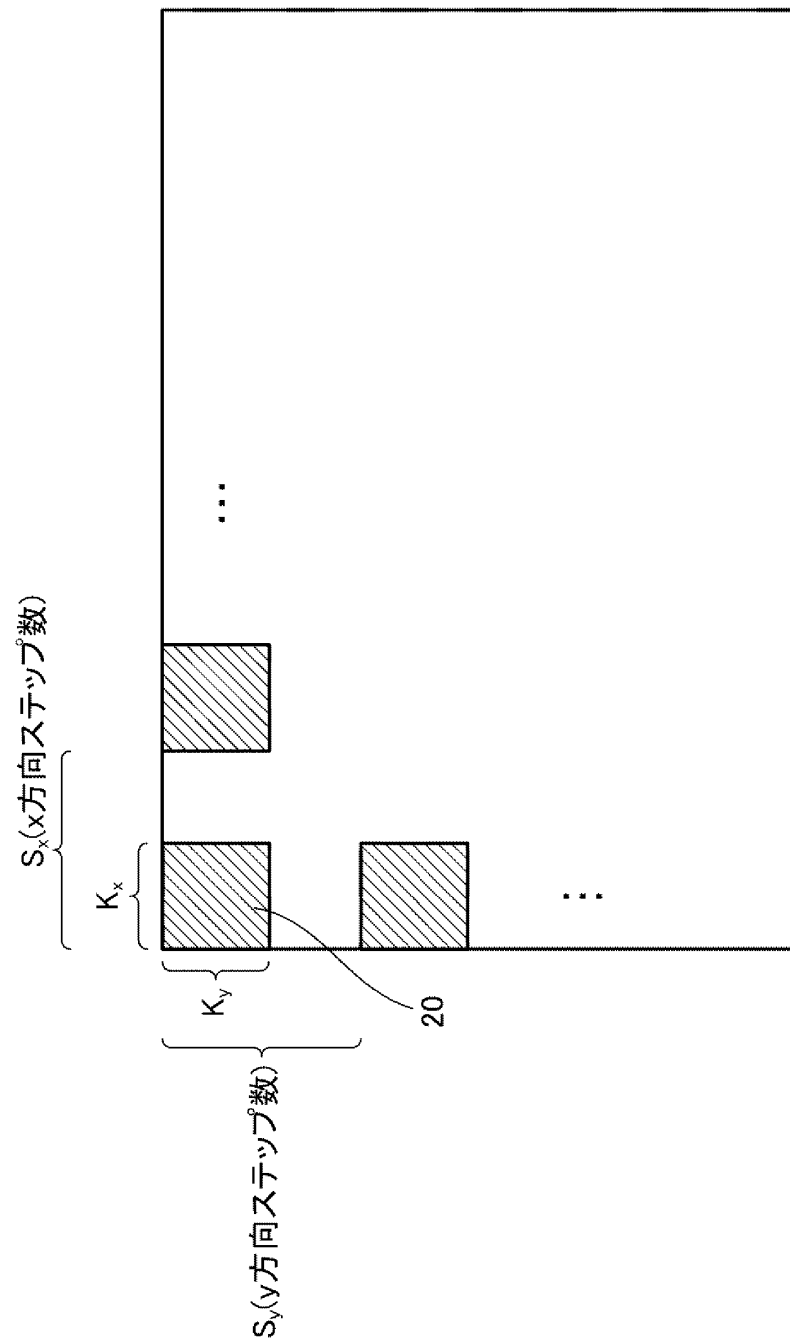
【図8】



[図9]



[図10]



[図11]

		ノードインデックス				
		1	2	3	...	N_l
エッジインデックス	1	w_{11}	w_{21}	w_{31}	...	w_{N_l1}
	2	w_{12}	w_{22}	w_{32}	...	w_{N_l2}
	3	w_{13}	w_{23}	w_{33}	...	w_{N_l3}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
	N_{l-1}	$w_{1N_{l-1}}$	$w_{2N_{l-1}}$	$w_{3N_{l-1}}$	...	$w_{N_lN_{l-1}}$
	$N_{l-1}+1$	$w_{1N_{l-1}+1}$	$w_{2N_{l-1}+1}$	$w_{3N_{l-1}+1}$	...	$w_{N_lN_{l-1}+1}$

[図12]

		ノードインデックス				
		1	2	3	...	N_l
エッジインデックス	1	q_{11}	q_{21}	q_{31}	...	q_{N_l1}
	2	q_{12}	q_{22}	q_{32}	...	q_{N_l2}
	3	q_{13}	q_{23}	q_{33}	...	q_{N_l3}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
	N_{l-1}	$q_{1N_{l-1}}$	$q_{2N_{l-1}}$	$q_{3N_{l-1}}$	...	$q_{N_lN_{l-1}}$
	$N_{l-1}+1$	$q_{1N_{l-1}+1}$	$q_{2N_{l-1}+1}$	$q_{3N_{l-1}+1}$	...	$q_{N_lN_{l-1}+1}$

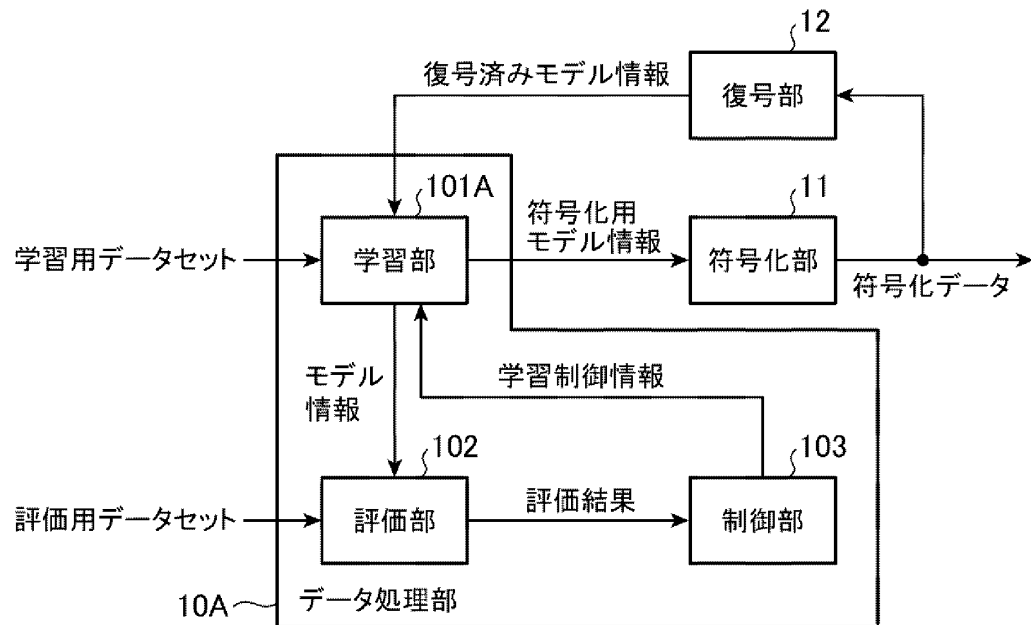
[図13]

		カーネルインデックス				
		1	2	3	...	M_l
エッジインデックス	1	w_{11}	w_{21}	w_{31}	...	w_{M_l1}
	2	w_{12}	w_{22}	w_{32}	...	w_{M_l2}
	3	w_{13}	w_{23}	w_{33}	...	w_{M_l3}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
	K_l	w_{1K_l}	w_{2K_l}	w_{3K_l}	...	$w_{M_lK_l}$
	K_l+1	w_{1K_l+1}	w_{2K_l+1}	w_{3K_l+1}	...	$w_{M_lK_l+1}$

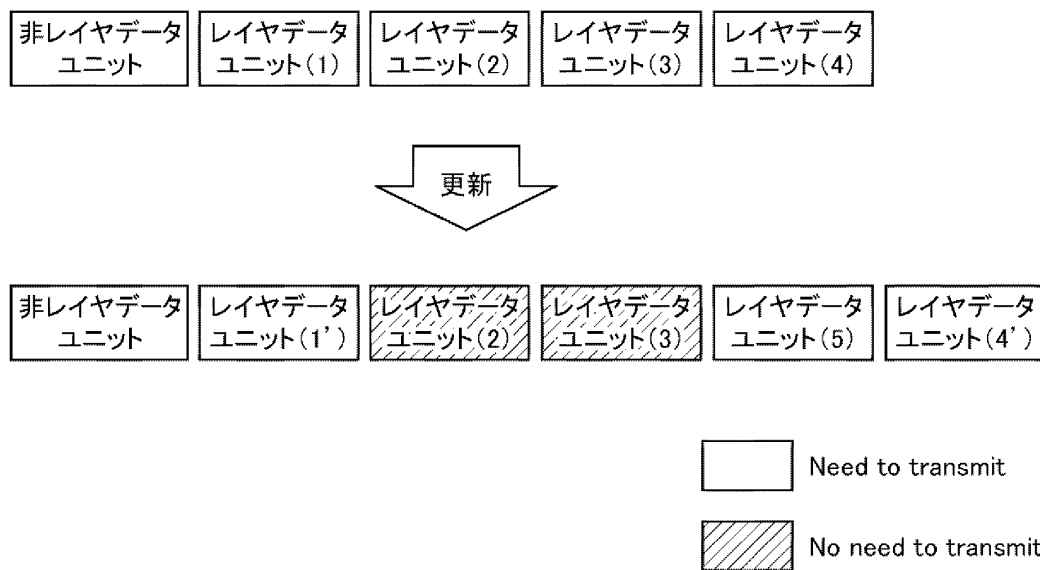
[図14]

		カーネルインデックス				
		1	2	3	...	M_l
エッジインデックス	1	q_{11}	q_{21}	q_{31}	...	q_{M_l1}
	2	q_{12}	q_{22}	q_{32}	...	q_{M_l2}
	3	q_{13}	q_{23}	q_{33}	...	q_{M_l3}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
	K_l	q_{1K_l}	q_{2K_l}	q_{3K_l}	...	$q_{M_lK_l}$
	K_l+1	q_{1K_l+1}	q_{2K_l+1}	q_{3K_l+1}	...	$q_{M_lK_l+1}$

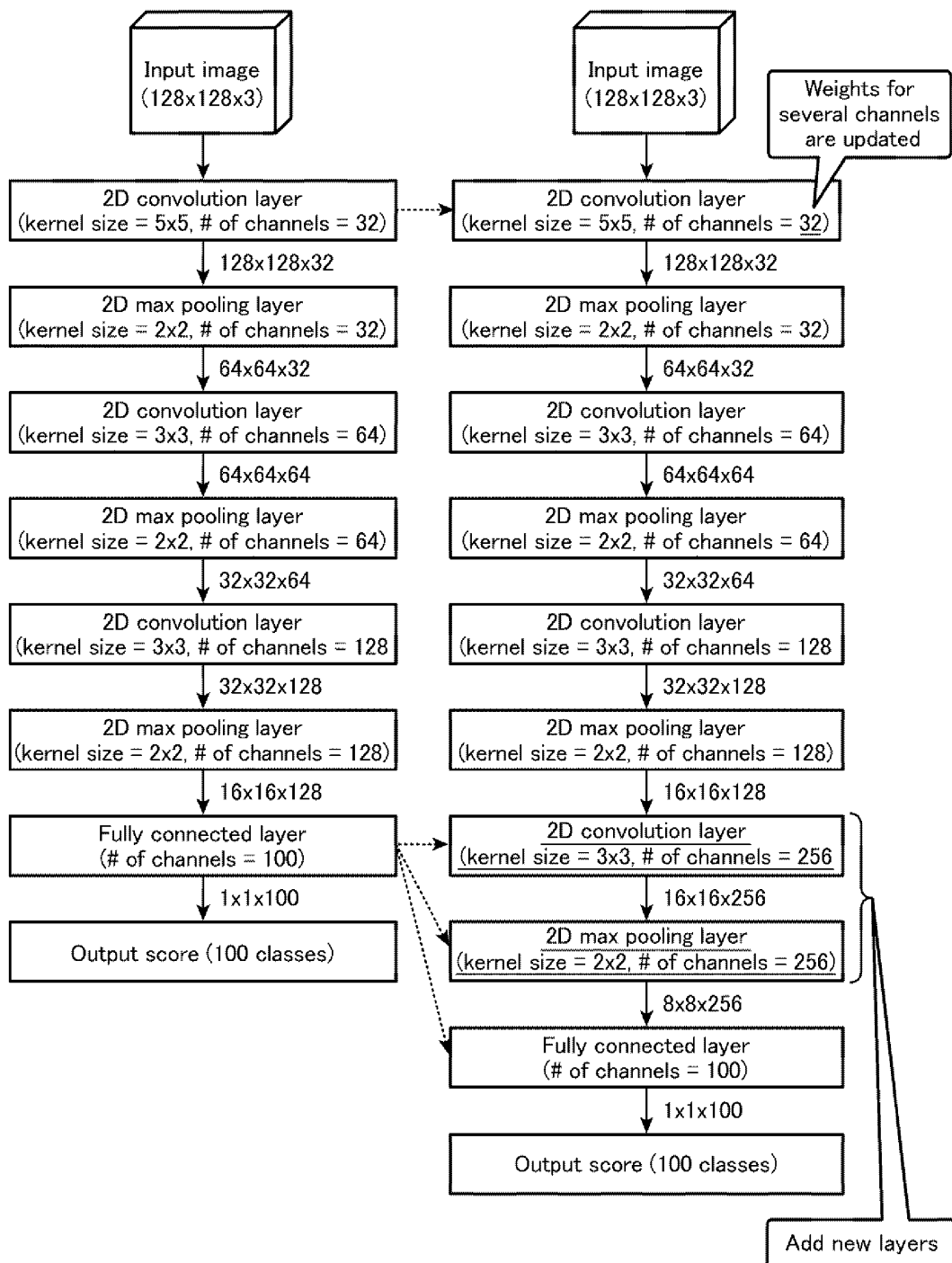
[図15]



[図16]



[図17]



[図18]

(B) model_id = 10のネットワークモデル

```

graph ExampleNet( input ) -> ( output )
{
    input = external(shape = [1, 3, 128, 128])
    kernel1 = variable(shape = [32, 3, 5, 5], label = 'exampleret/conv1/kernel')
    bias1 = variable(shape = [1, 32], label = 'exampleret/conv1/bias')
    conv1 = conv(input, kernel1, bias1, padding = [(1,1), (1,1)],
        border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu1 = relu(conv1)
    pool1 = max_pool(relu1, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
        border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel2 = variable(shape = [64, 32, 3, 3], label = 'exampleret/conv2/kernel')
    bias2 = variable(shape = [1, 64], label = 'exampleret/conv2/bias')
    conv2 = conv(pool1, kernel2, bias2, padding = [(1,1), (1,1)],
        border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu2 = relu(conv2)
    pool2 = max_pool(relu2, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
        border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel3 = variable(shape = [128, 64, 3, 3], label = 'exampleret/conv3/kernel')
    bias3 = variable(shape = [1, 128], label = 'exampleret/conv3/bias')
    conv3 = conv(pool2, kernel3, bias3, padding = [(1,1), (1,1)],
        border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu3 = relu(conv3)
    pool3 = max_pool(relu3, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
        border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel4 = variable(shape = [256, 128, 3, 3], label = 'exampleret/conv4/kernel')
    bias4 = variable(shape = [1, 256], label = 'exampleret/conv4/bias')
    conv4 = conv(pool3, kernel4, bias4, padding = [(1,1), (1,1)],
        border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu4 = relu(conv4)
    pool4 = max_pool(relu4, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
        border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel5 = variable(shape = [100, 256, 8, 8], label = 'exampleret/fc5/kernel')
    bias5 = variable(shape = [1, 100], label = 'exampleret/fc5/bias')
    conv5 = conv(pool4, kernel5, bias5, padding = [(0,0), (0,0)],
        border = 'constant', stride = [1, 1], dilation = [1, 1])
    output = softmax(conv5)
}

```

(A) model_id = 00のネットワークモデル

```

graph ExampleNet( input ) -> ( output )
{
    input = external(shape = [1, 3, 128, 128])
    kernel1 = variable(shape = [32, 3, 5, 5], label = 'exampleret/conv1/kernel')
    bias1 = variable(shape = [1, 32], label = 'exampleret/conv1/bias')
    conv1 = conv(input, kernel1, bias1, padding = [(1,1), (1,1)],
        border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu1 = relu(conv1)
    pool1 = max_pool(relu1, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
        border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel2 = variable(shape = [64, 32, 3, 3], label = 'exampleret/conv2/kernel')
    bias2 = variable(shape = [1, 64], label = 'exampleret/conv2/bias')
    conv2 = conv(pool1, kernel2, bias2, padding = [(1,1), (1,1)],
        border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu2 = relu(conv2)
    pool2 = max_pool(relu2, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
        border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel3 = variable(shape = [128, 64, 3, 3], label =
'exampleret/conv3/kernel')
    bias3 = variable(shape = [1, 128], label = 'exampleret/conv3/bias')
    conv3 = conv(pool2, kernel3, bias3, padding = [(1,1), (1,1)],
        border = 'constant', stride = [1, 1], dilation = [1, 1])
    relu3 = relu(conv3)
    pool3 = max_pool(relu3, size = [1, 1, 2, 2], stride = [1, 1, 2, 2],
        border = 'ignore', padding = [(0,0), (0,0), (0,0), (0,0)])
    kernel4 = variable(shape = [100, 128, 16, 16], label =
'exampleret/fc4/kernel')
    bias4 = variable(shape = [1, 100], label = 'exampleret/fc4/bias')
    conv4 = conv(pool3, kernel4, bias4, padding = [(0,0), (0,0)],
        border = 'constant', stride = [1, 1], dilation = [1, 1])
    output = softmax(conv4)
}

```

[図19]

(A) model_id = 0のネットワークモデル

```
layer ExampleNet( input ) -> ( output )  
{  
  0 = kernel1, bias1  
  1 = kernel2, bias2  
  2 = kernel3, bias3  
  3 = kernel4, bias4  
}
```

(B) model_id = 10のネットワークモデル

```
layer ExampleNet( input ) -> ( output )  
{  
  0 = kernel1, bias1  
  1 = kernel2, bias2  
  2 = kernel3, bias3  
  3 = kernel5, bias5  
  4 = kernel4, bias4  
}
```

[図20]

図20A

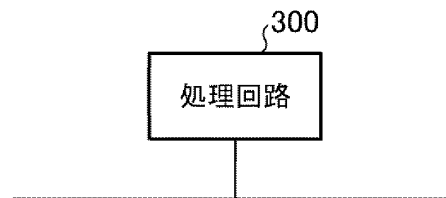


図20B

