Development of C programs for Convolutional Neural Network Accelerators

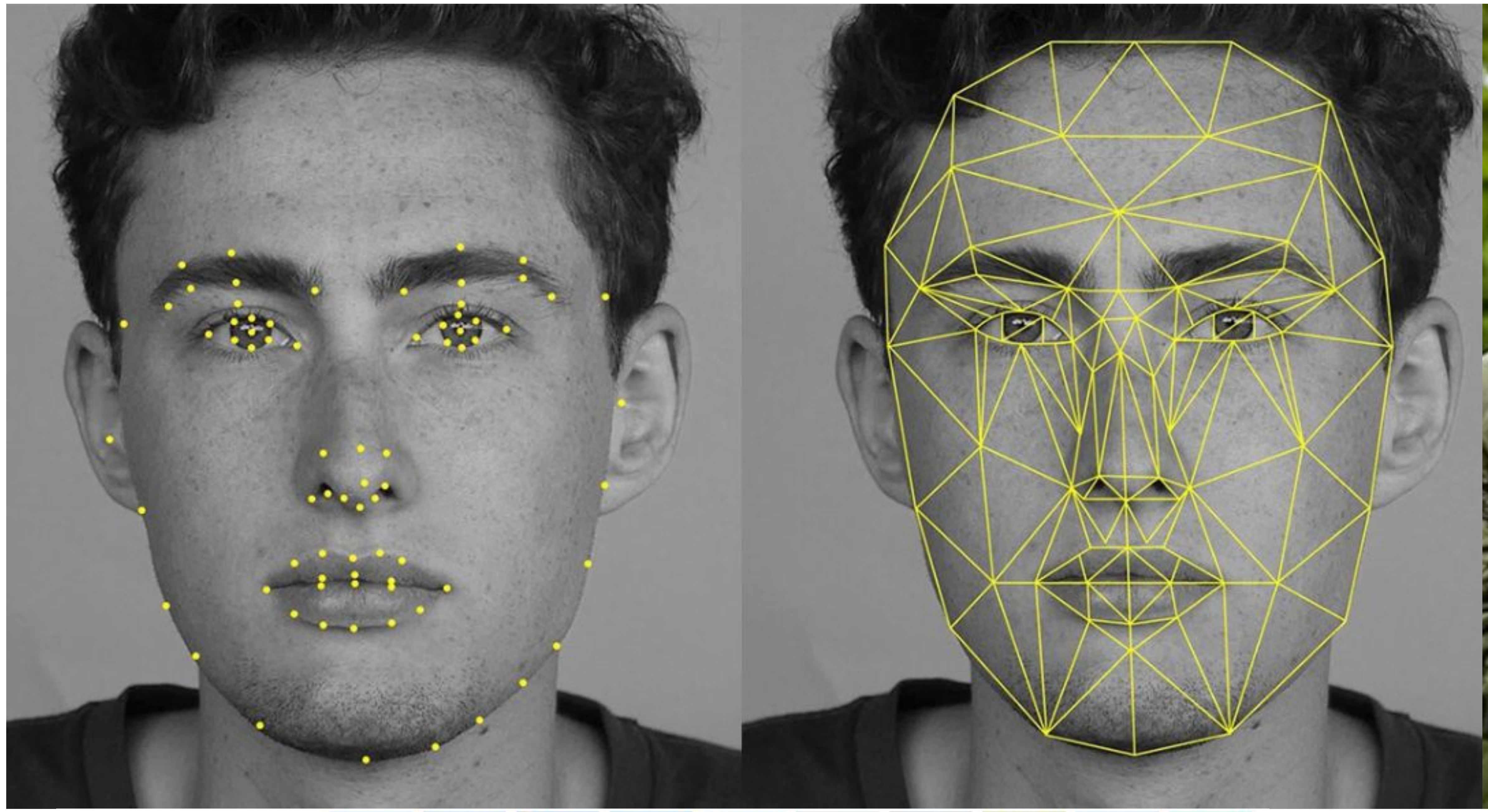
JiYoung An, Sujin Kang
Prof. Nikil Dutt , Kenshu Seto, Hamid Nejatollahi

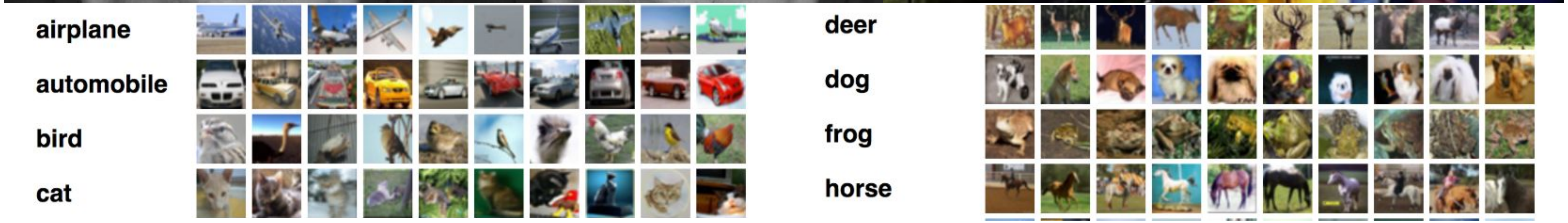
Dept. of Computer Engineering in Kyung Hee University
Dept. of Computer Engineering in Hanyang University
University of California Irvine
University in Tokyo

BACKGROUND

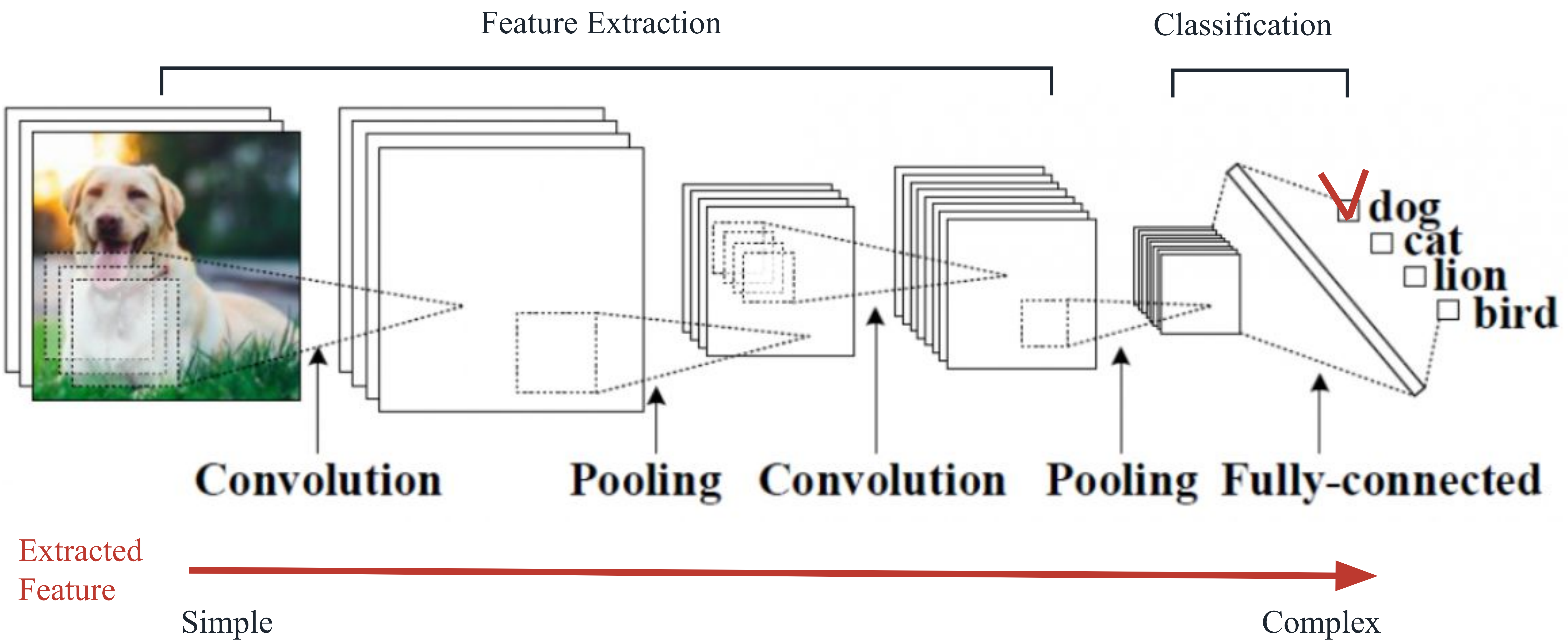
Convolution Neural Network

Keyword : Massive data , Image recognition and classification





1. Convolutional Neural Networks



Feature Extraction

Classification

Extracted Feature

Simple

Complex

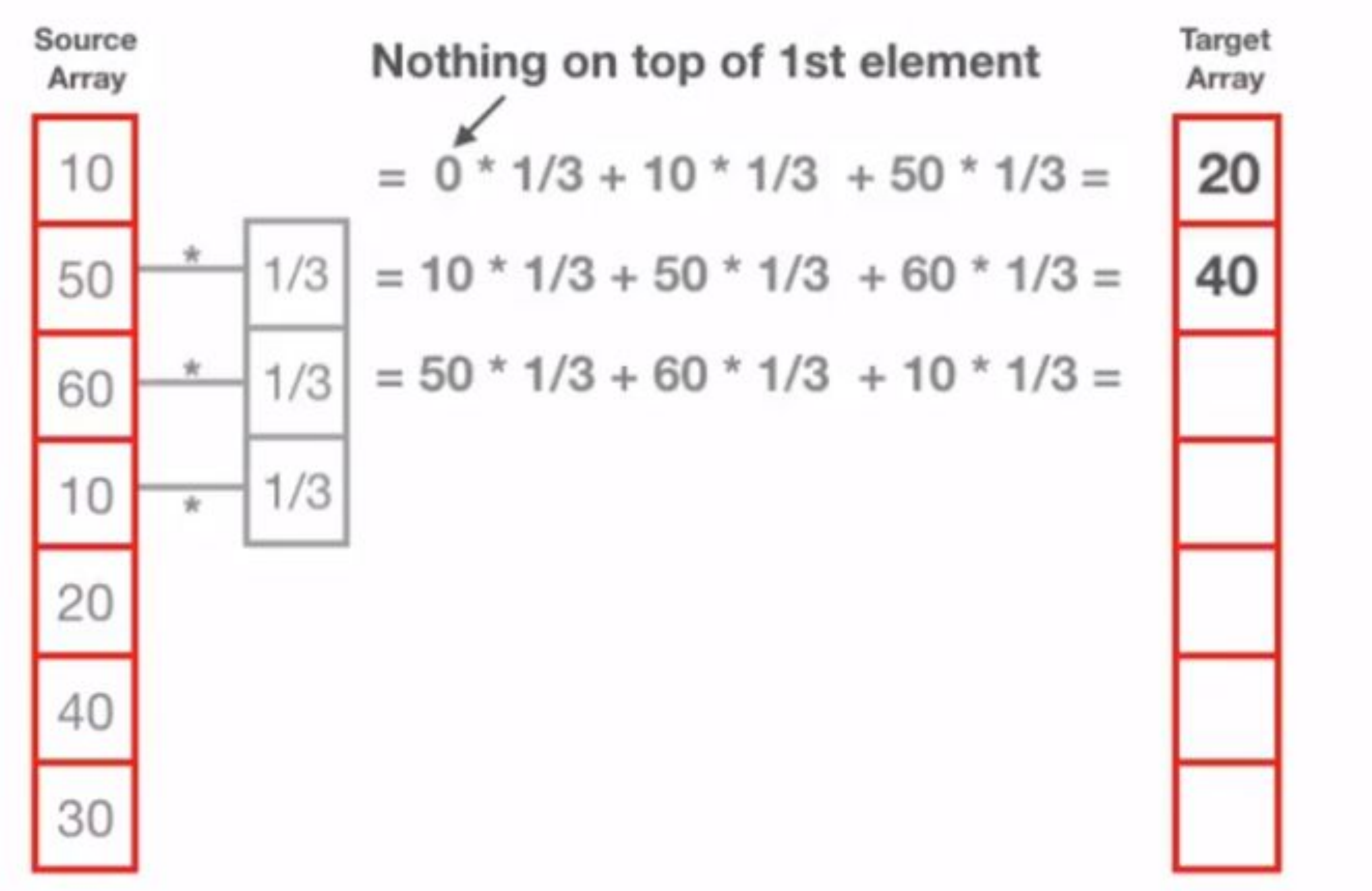
Keep spatial Information of Image

Automatically extract and learn its features

https://www.inflearn.com/course-status-2/

2. Convolution layer

Output = Source * Filter (* Convolution)



Nothing on top of 1st element

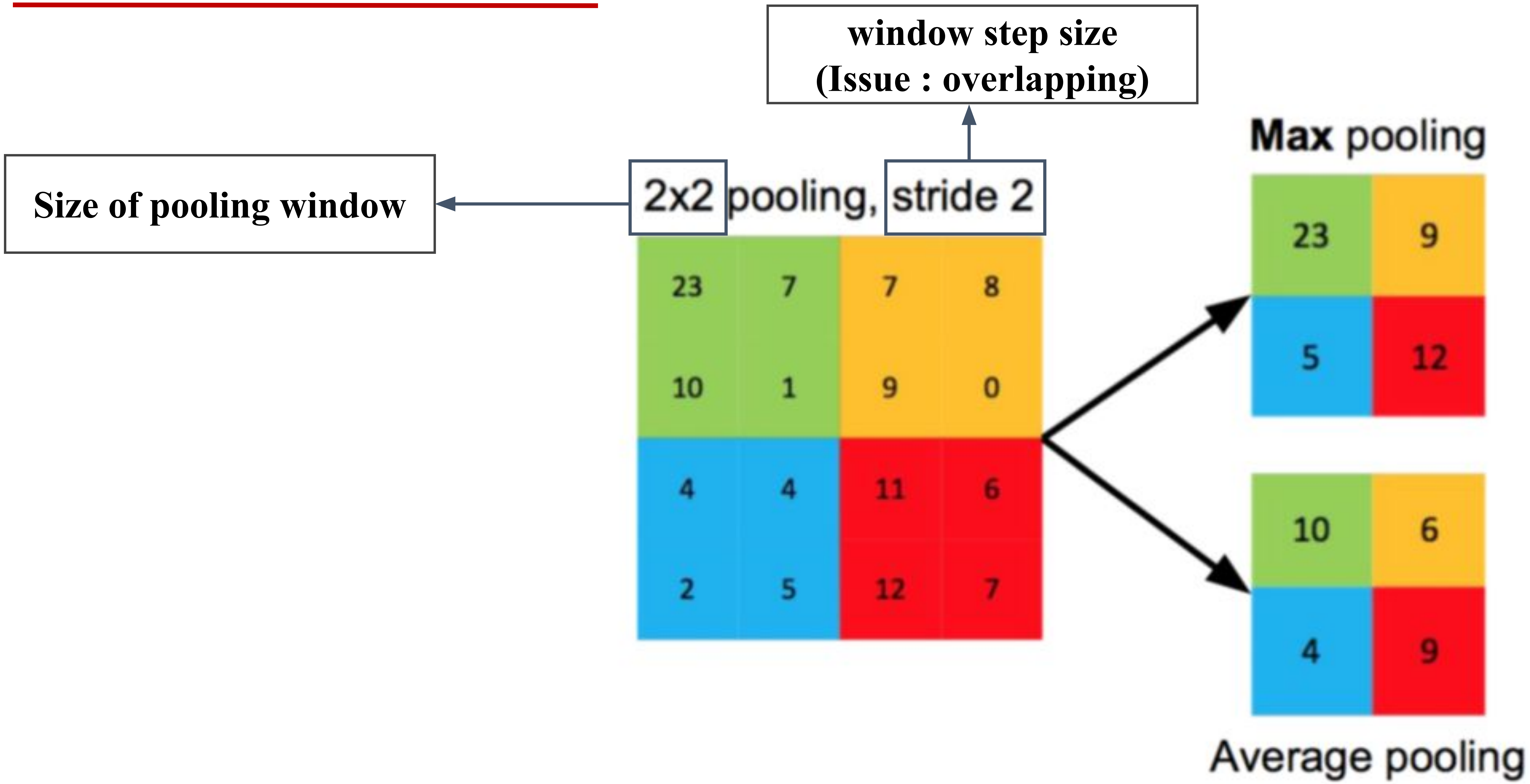
= 0 * 1/3 + 10 * 1/3 + 50 * 1/3 =

= 10 * 1/3 + 50 * 1/3 + 60 * 1/3 =

= 50 * 1/3 + 60 * 1/3 + 10 * 1/3 =

https://www.youtube.com/watch?v=ulKbLD6BRJA

3. Pooling layer

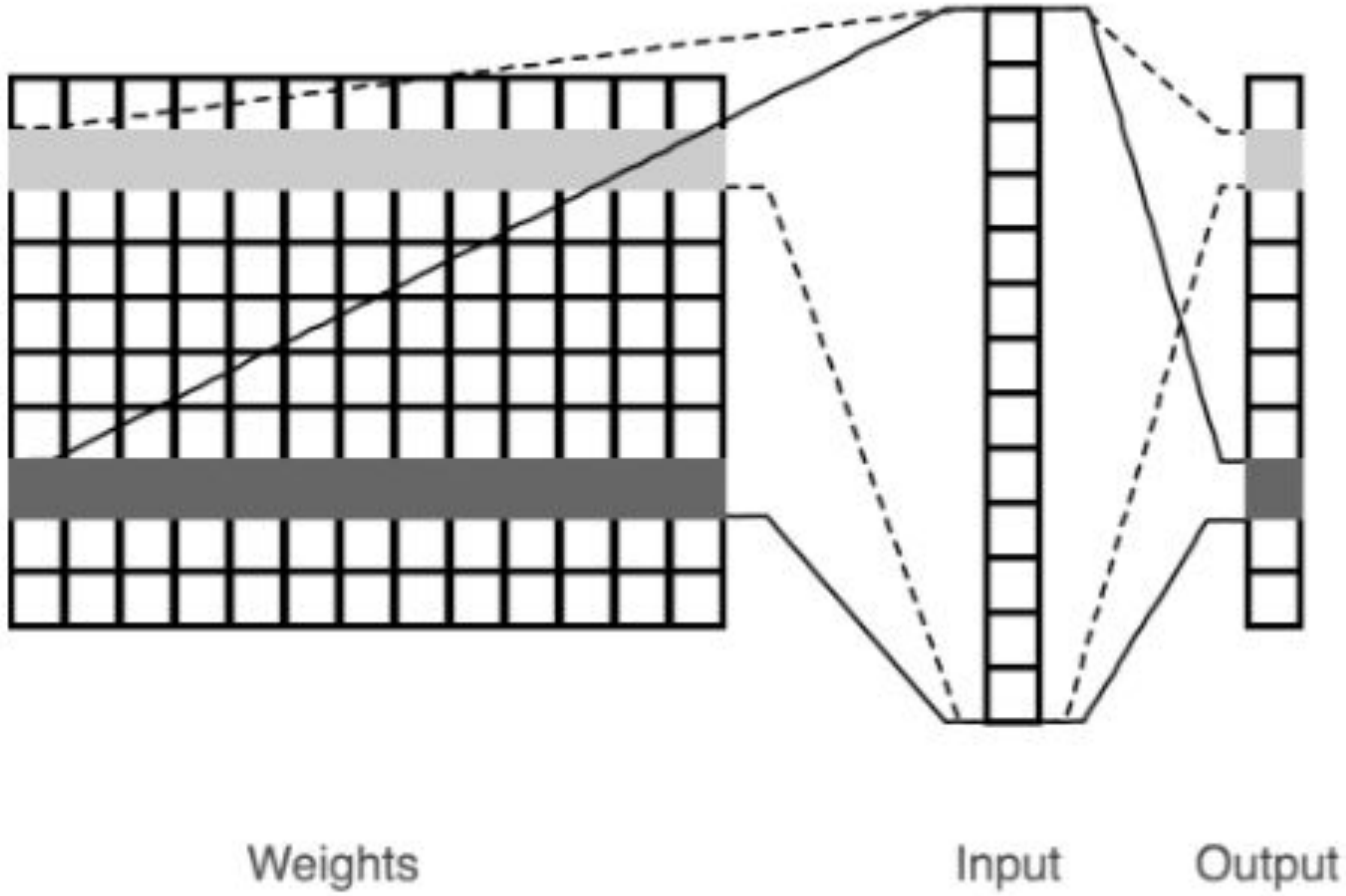


Size of pooling window

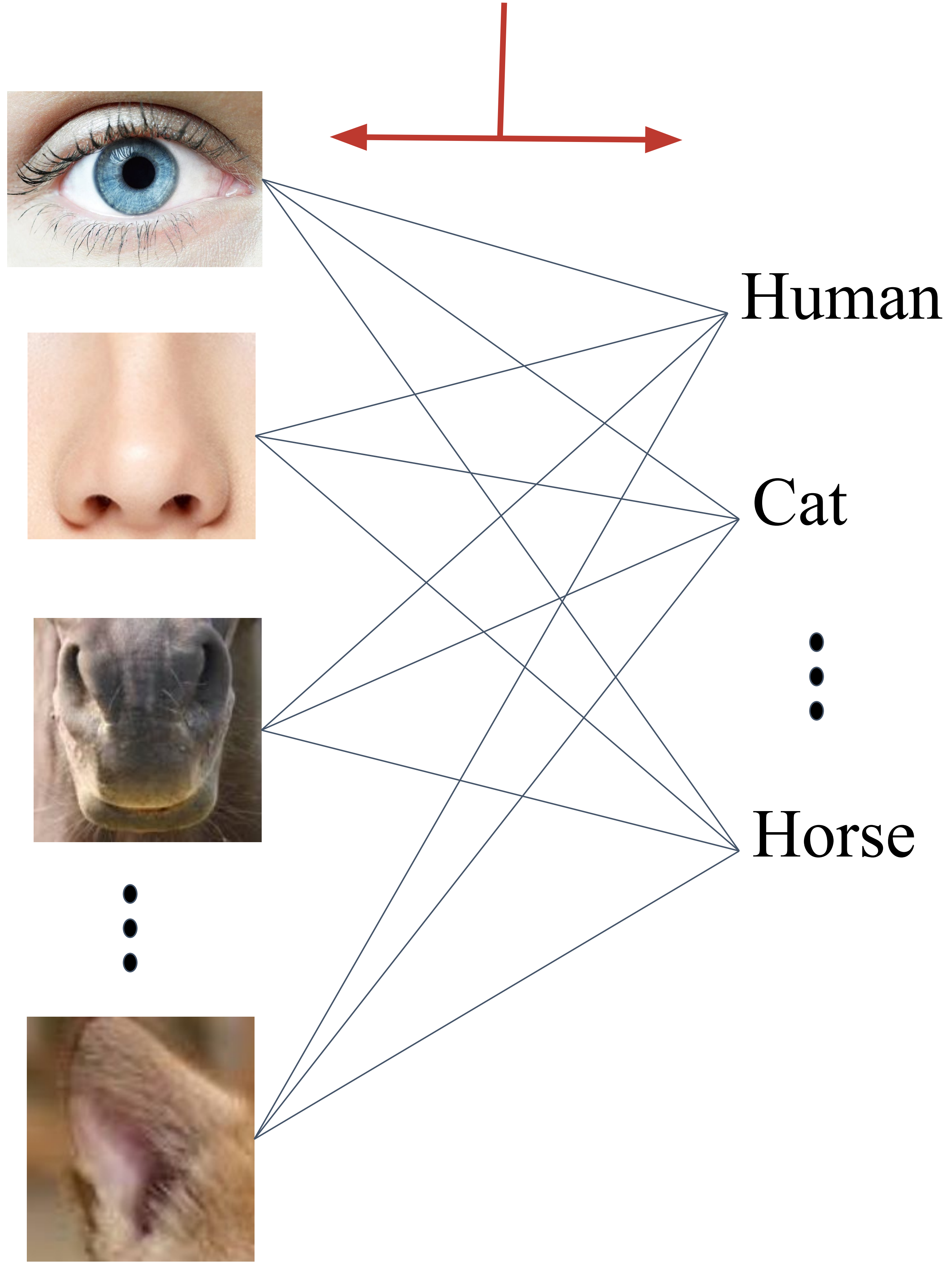
window step size
(Issue : overlapping)

1. Extract strong feature and reduce resolution
2. Increase translation-invariance and noise-resilience

4. Fully connected layer



The more Complex, the deeper it becomes



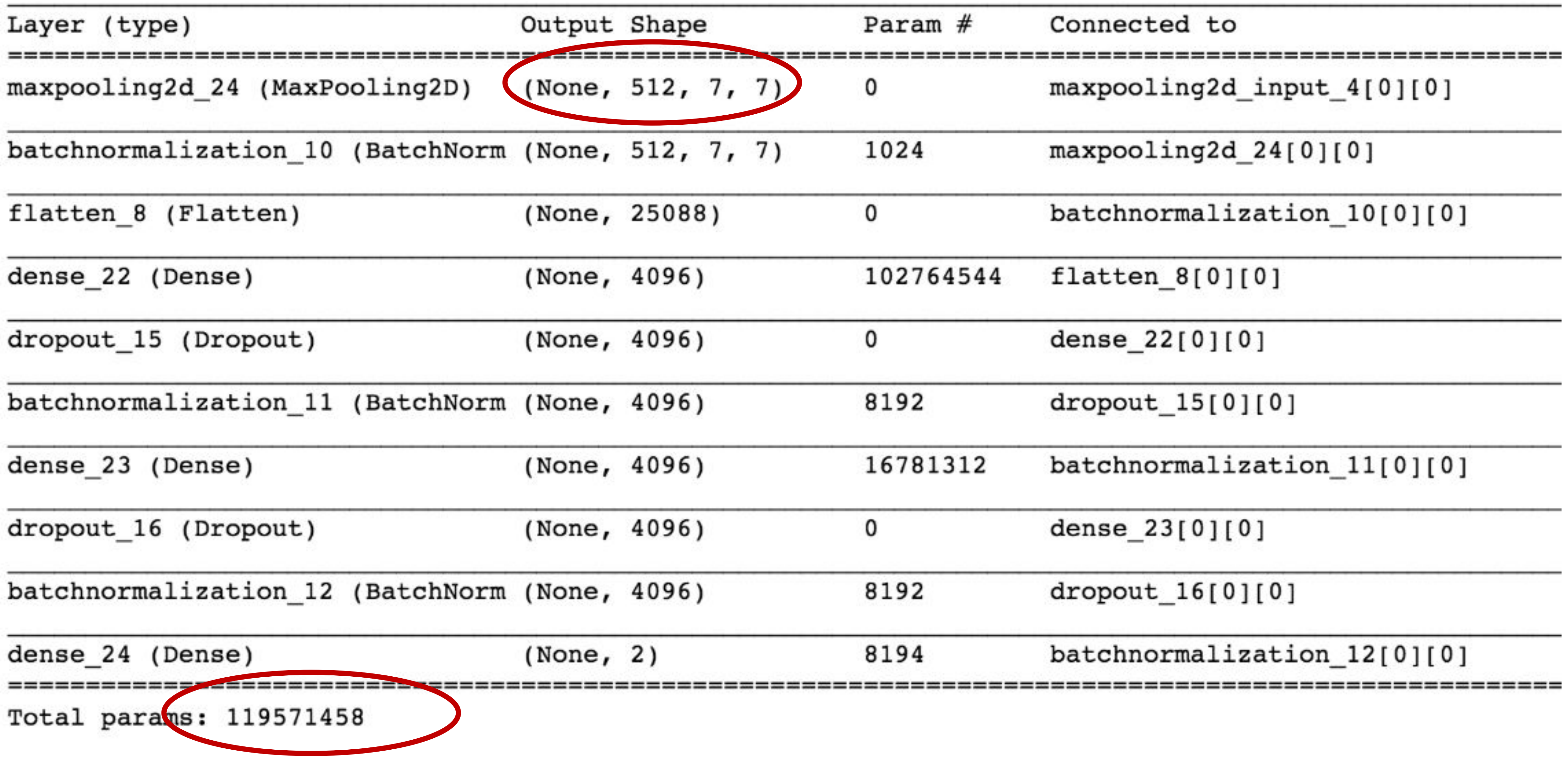
With Extracted Features, do classification

https://www.safaribooksonline.com/library/view/machine-learning-with/9781787121515/79694e8a-e30d-4b30-9b59-abcb2c436182.xhtml

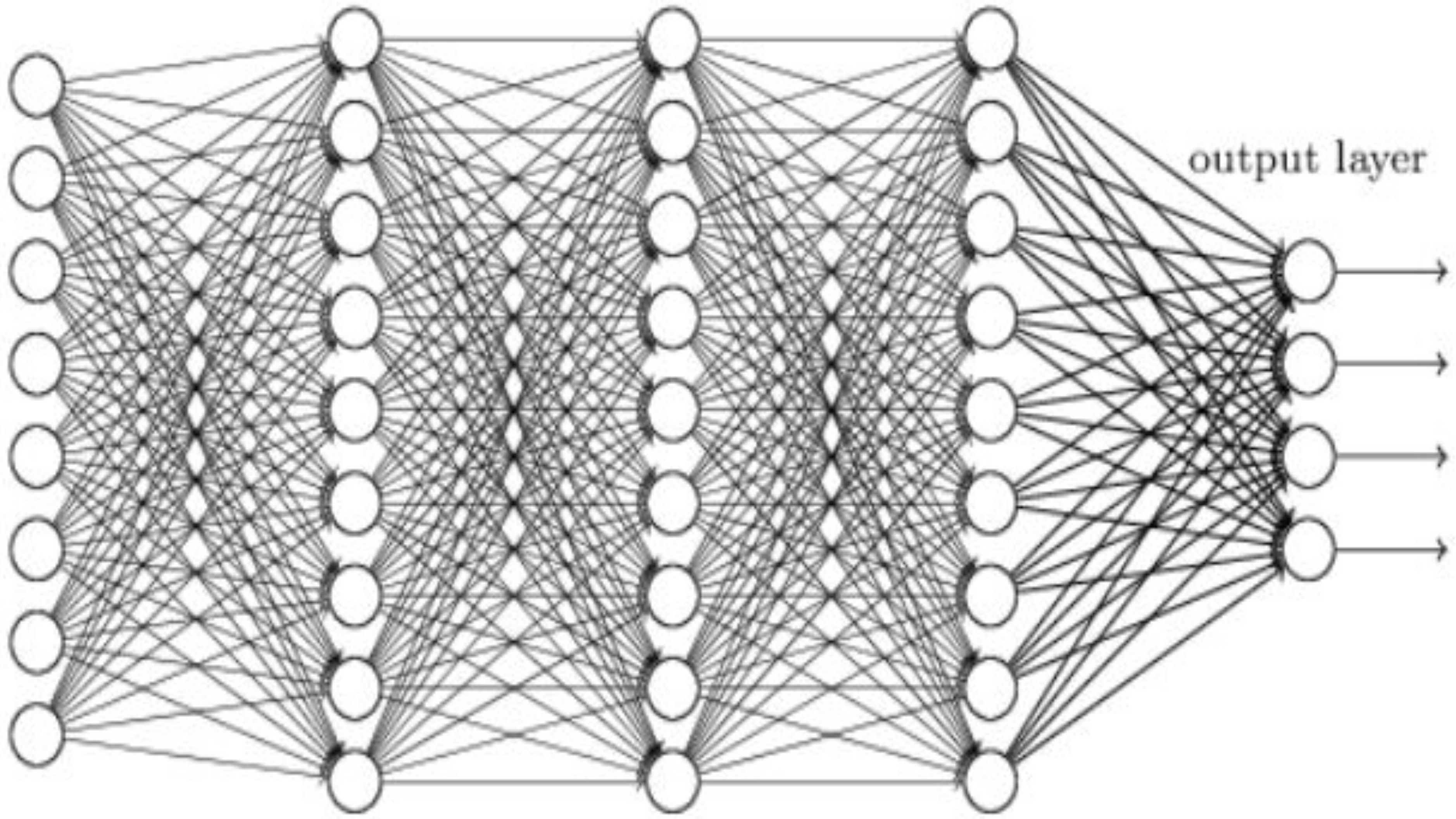
PROJECT

Problem of CNNs

Keyword : Time-consuming, Computing Energy, Calculation



```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
maxpooling2d_24 (MaxPooling2D)   (None, 512, 7, 7)     0           maxpooling2d_input_4[0][0]
____________________________________________________________________________________________________
batchnormalization_10 (BatchNorm (None, 512, 7, 7)     1024        maxpooling2d_24[0][0]
____________________________________________________________________________________________________
flatten_8 (Flatten)              (None, 25088)         0           batchnormalization_10[0][0]
____________________________________________________________________________________________________
dense_22 (Dense)                 (None, 4096)          102764544   flatten_8[0][0]
____________________________________________________________________________________________________
dropout_15 (Dropout)             (None, 4096)          0           dense_22[0][0]
____________________________________________________________________________________________________
batchnormalization_11 (BatchNorm (None, 4096)          8192        dropout_15[0][0]
____________________________________________________________________________________________________
dense_23 (Dense)                 (None, 4096)          16781312    batchnormalization_11[0][0]
____________________________________________________________________________________________________
dropout_16 (Dropout)             (None, 4096)          0           dense_23[0][0]
____________________________________________________________________________________________________
batchnormalization_12 (BatchNorm (None, 4096)          8192        dropout_16[0][0]
____________________________________________________________________________________________________
dense_24 (Dense)                 (None, 2)             8194        batchnormalization_12[0][0]
====================================================================================================
Total params: 119571458
```

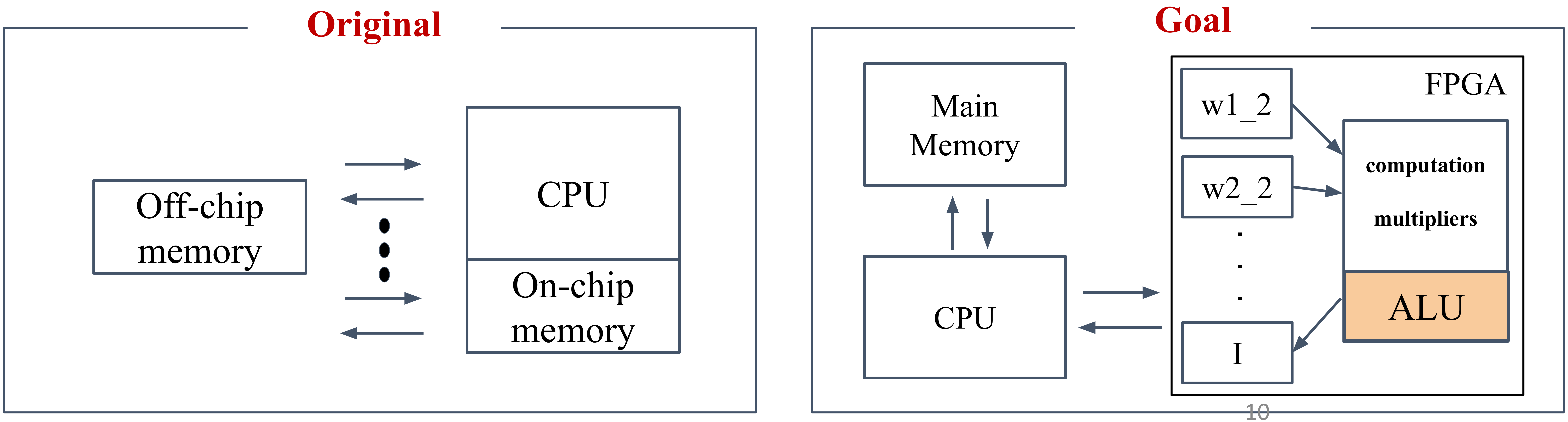


Required a lot of Computing resources and time

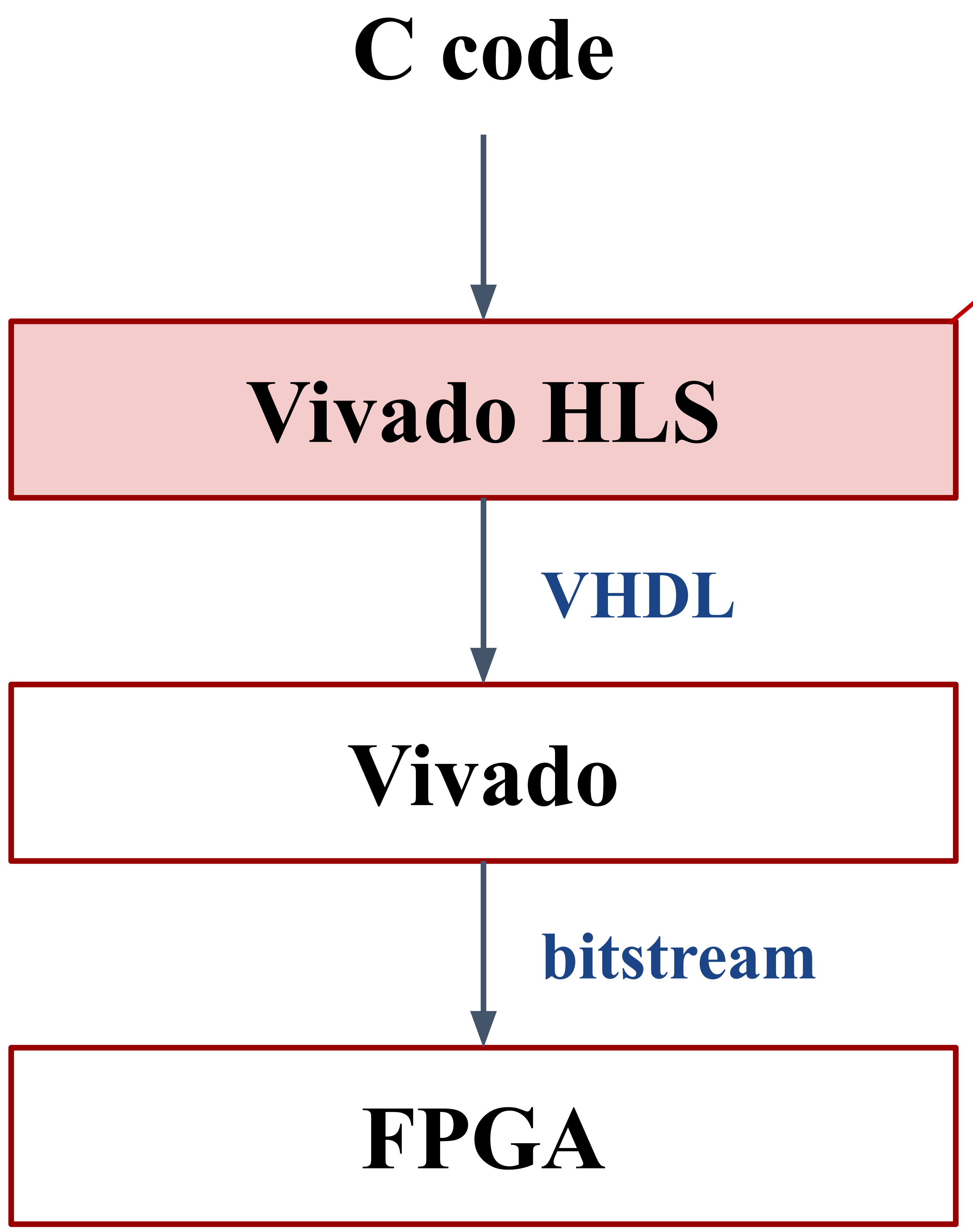
Purpose

Acceleration of Convolution Neural Networks to embed in light machine

Focus on Hardware Acceleration



Approach



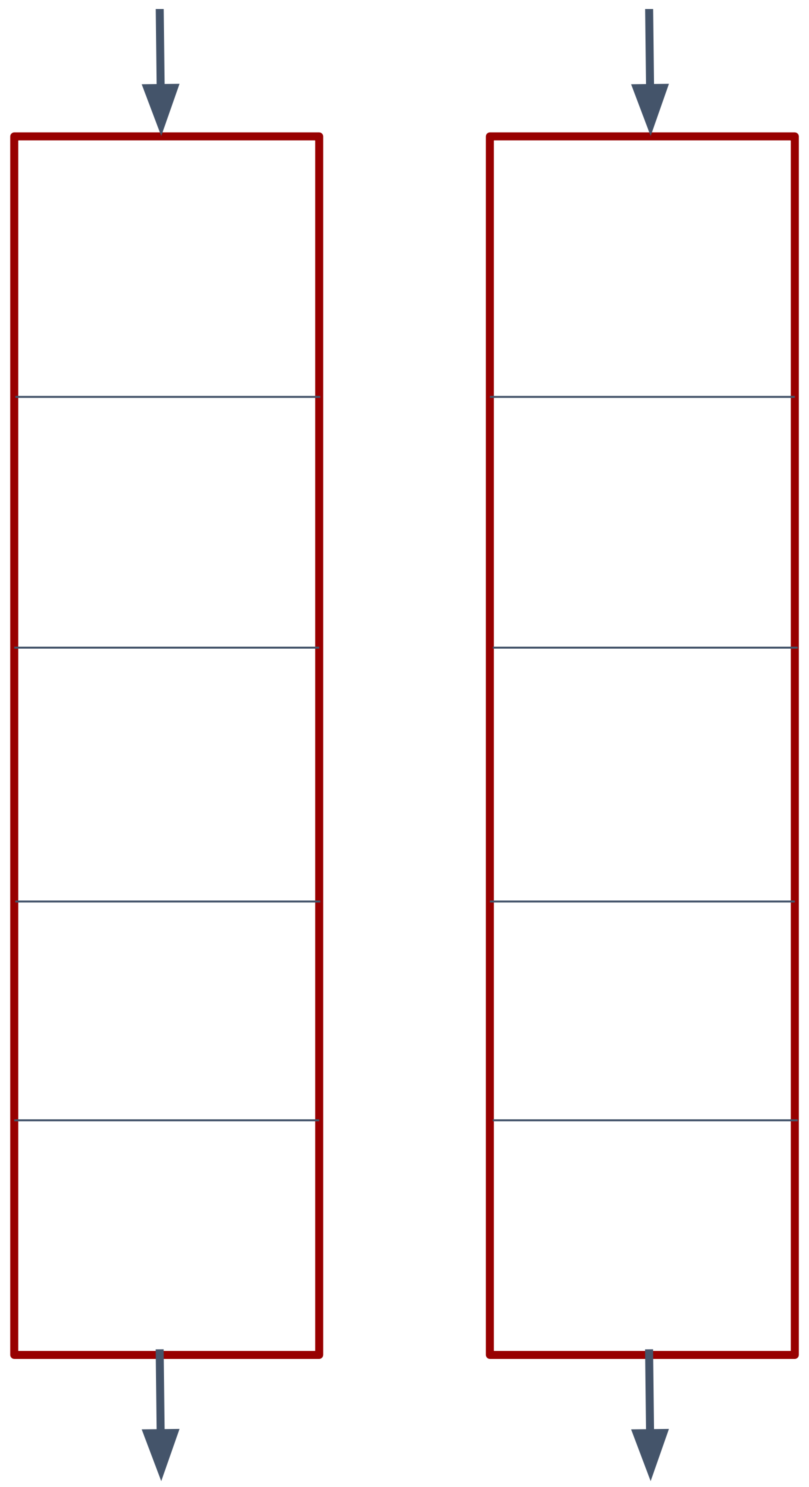
Keypoint

<u>Optimize Hardware</u> so that

- maximize computational efficiency
- minimize computational latency
- minimize memory access

1. Hardware Optimization : Array partition

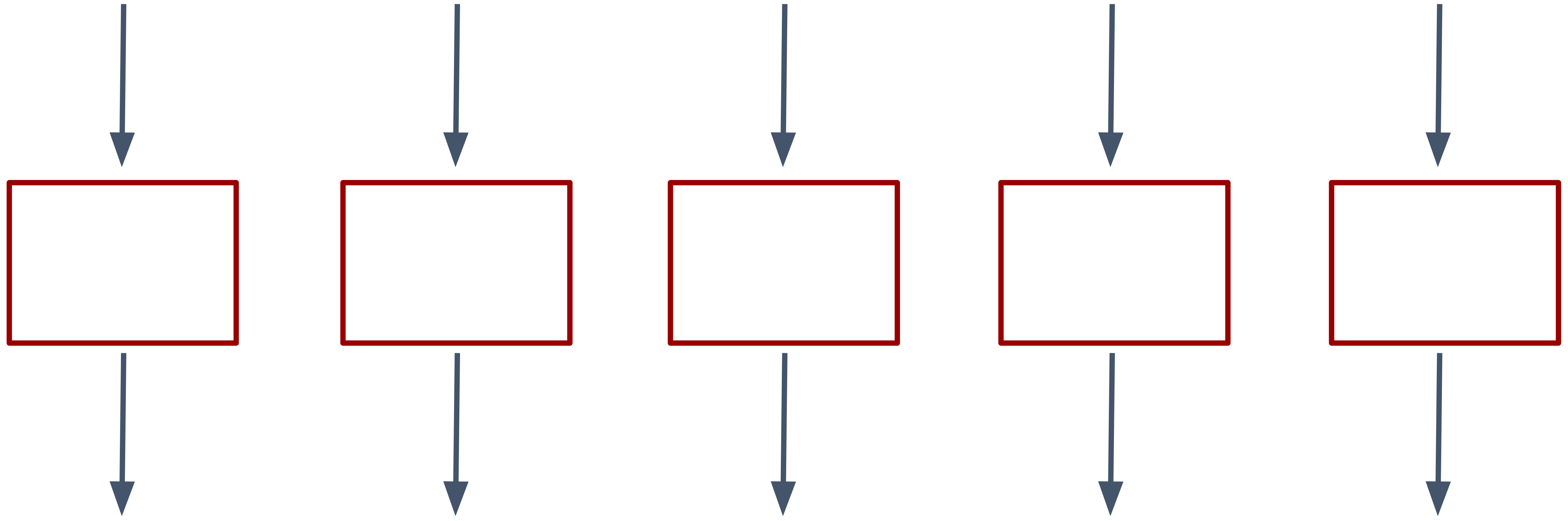
Normal



Array partition : Complete

Command Ex :

#pragma HLS ARRAY_PARTITION variable=B complete



Pros and Cons : Parallel access vs Hardware delay

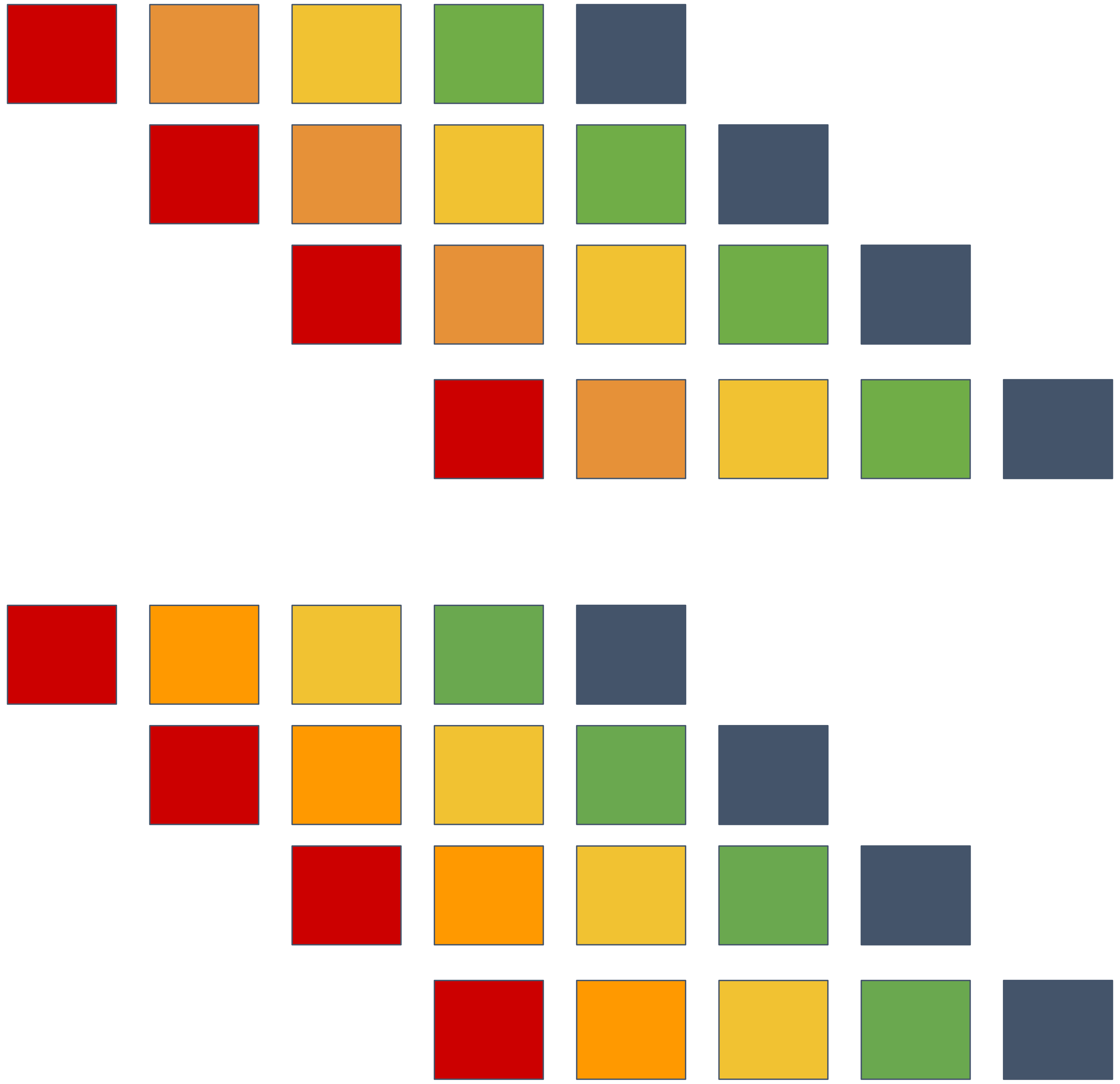
2. Hardware Optimization : Pipeline

In Convolution Code…

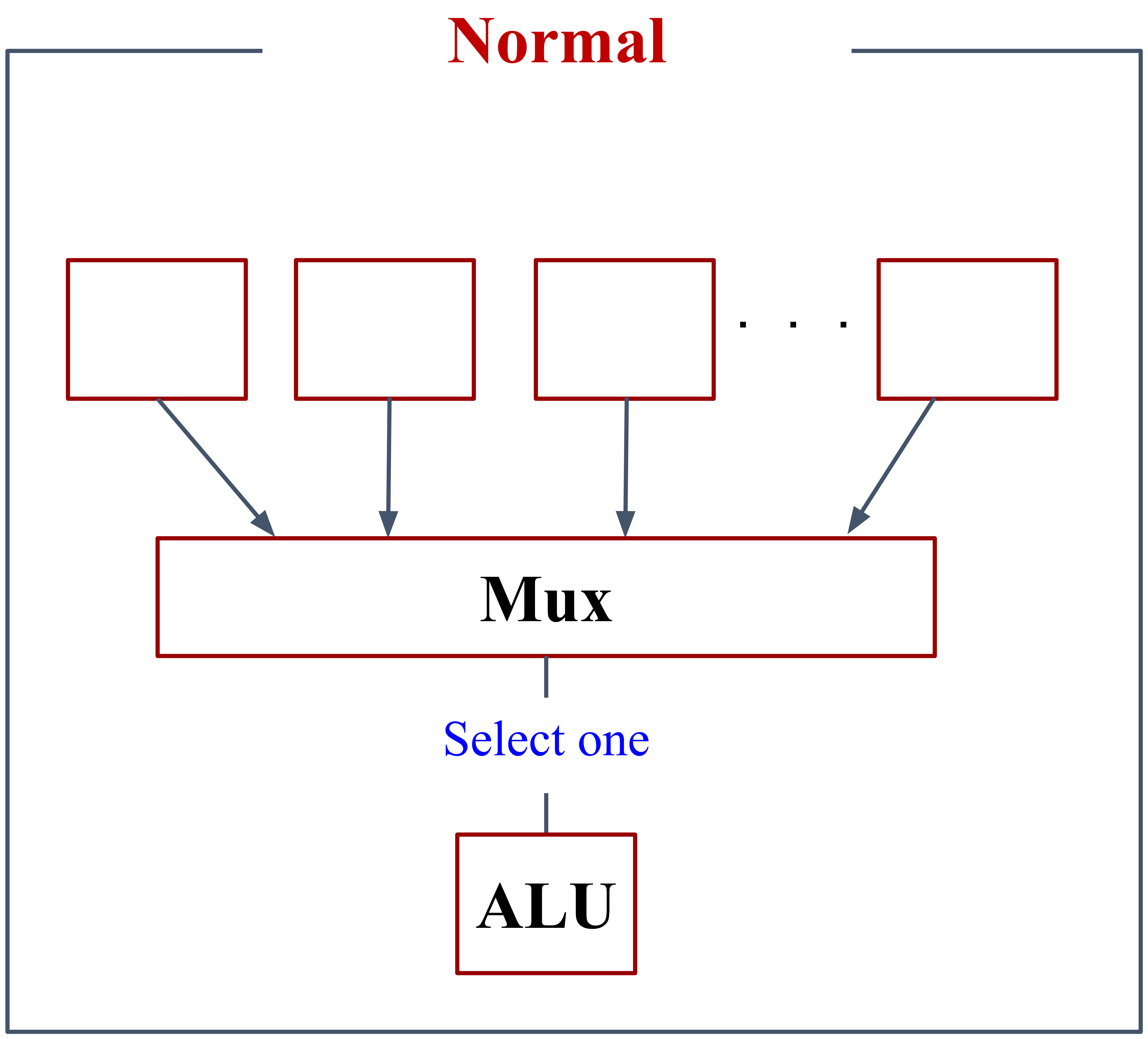
```
for( m = 0 ; m< 4096 ; m ++)
   for( n = 0 ; n < 2048 ; n ++ )
      for( i = 0 ; i < 256 ; i ++ )
         for( j = 0 ; j < 256 ; j ++ ){
            for(x =0 ; x <3 ; x ++){
               for(y = 0 ; y <3 ; y ++ ){
                  O[i][j]  += I[i][j] * W[m][n][x][y]
               }
            }
         }
```

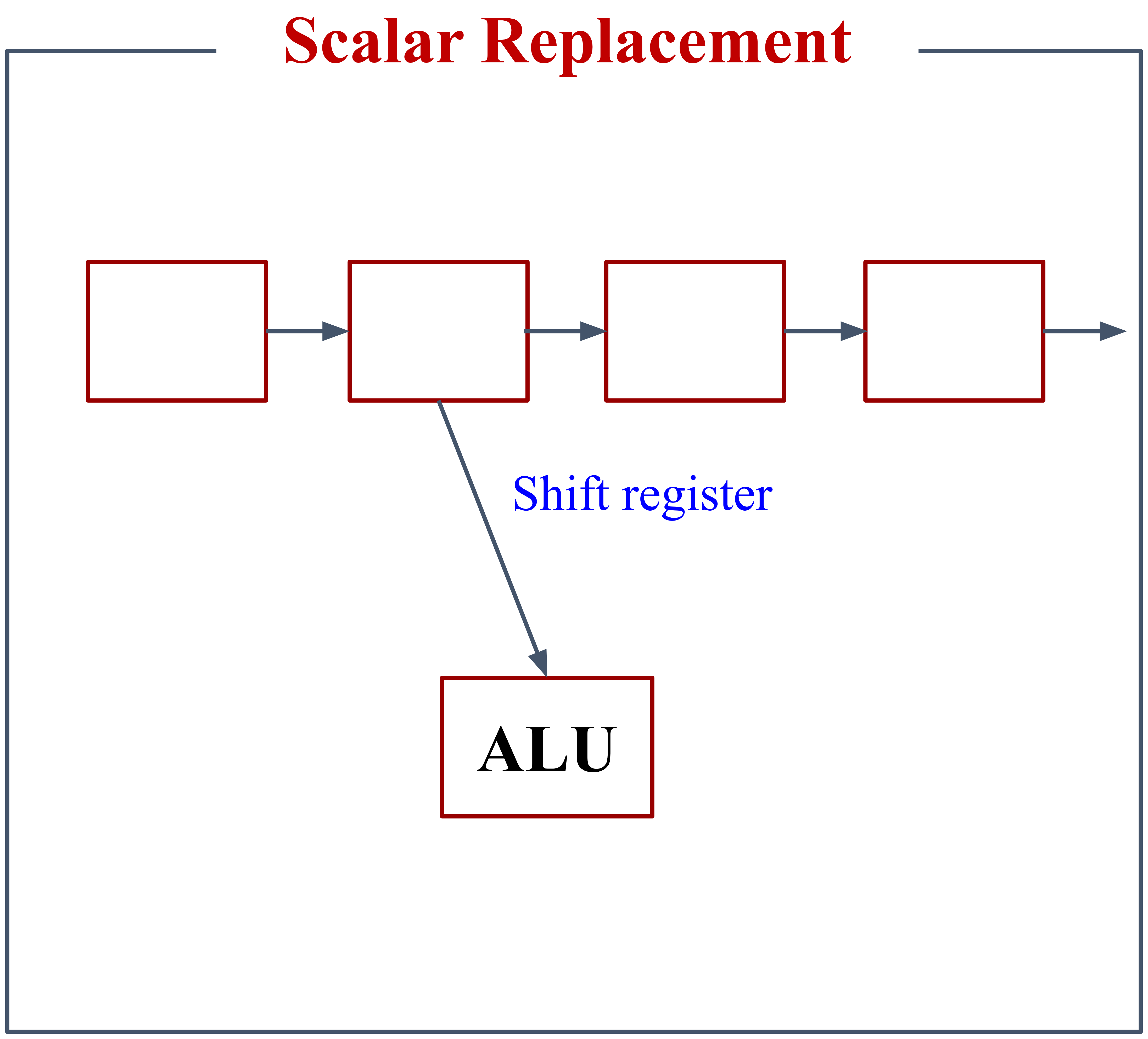
Apply Pipeline →

Command Ex : #pragma HLS PIPELINE



3. (Future)Hardware Optimization: Scalar Replacement





Our Role

1. Template library

Implement function
in CNN model
without model dependency

2. Verifier

SW version
vs
Python(keras)

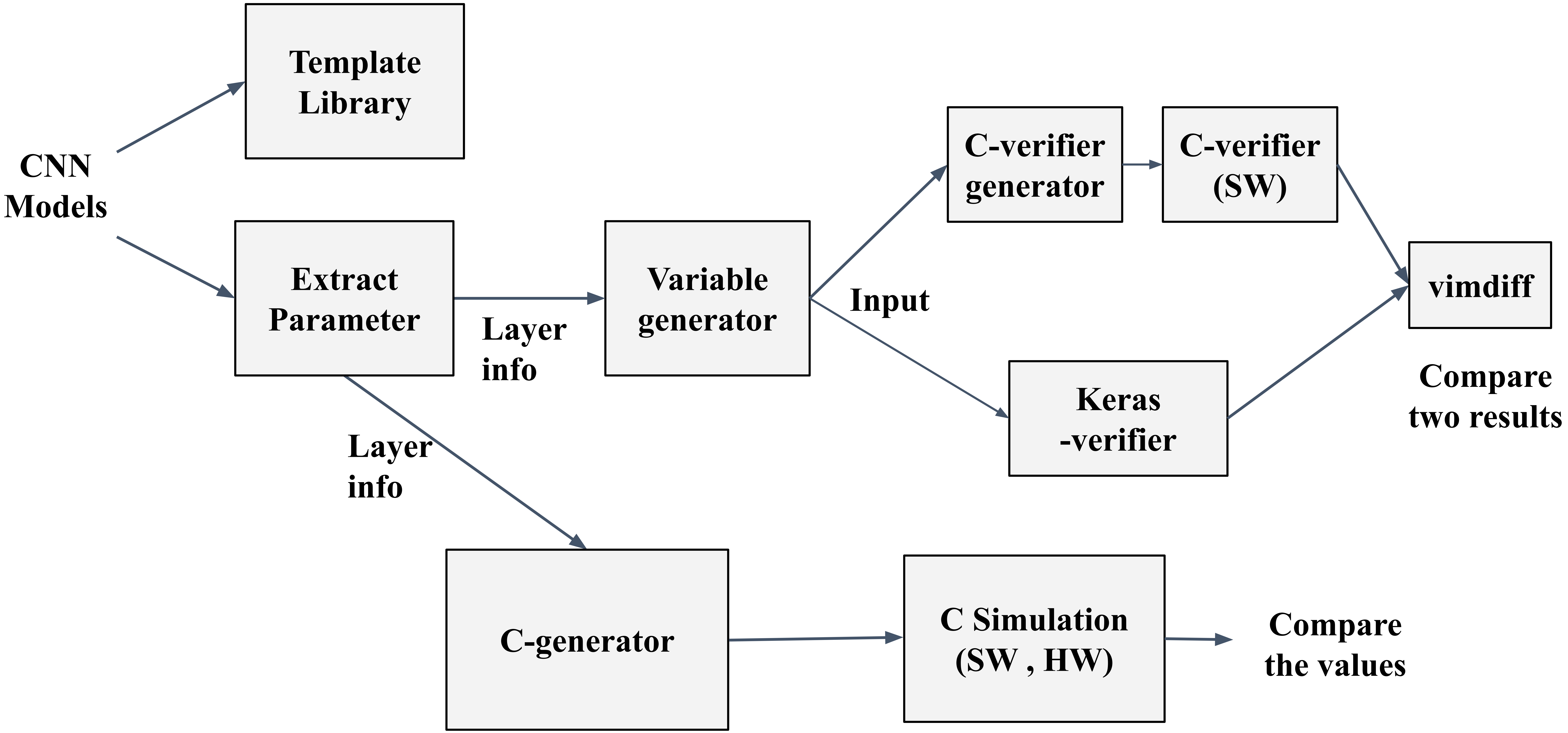
3. Generator

Generate C code for Vivado

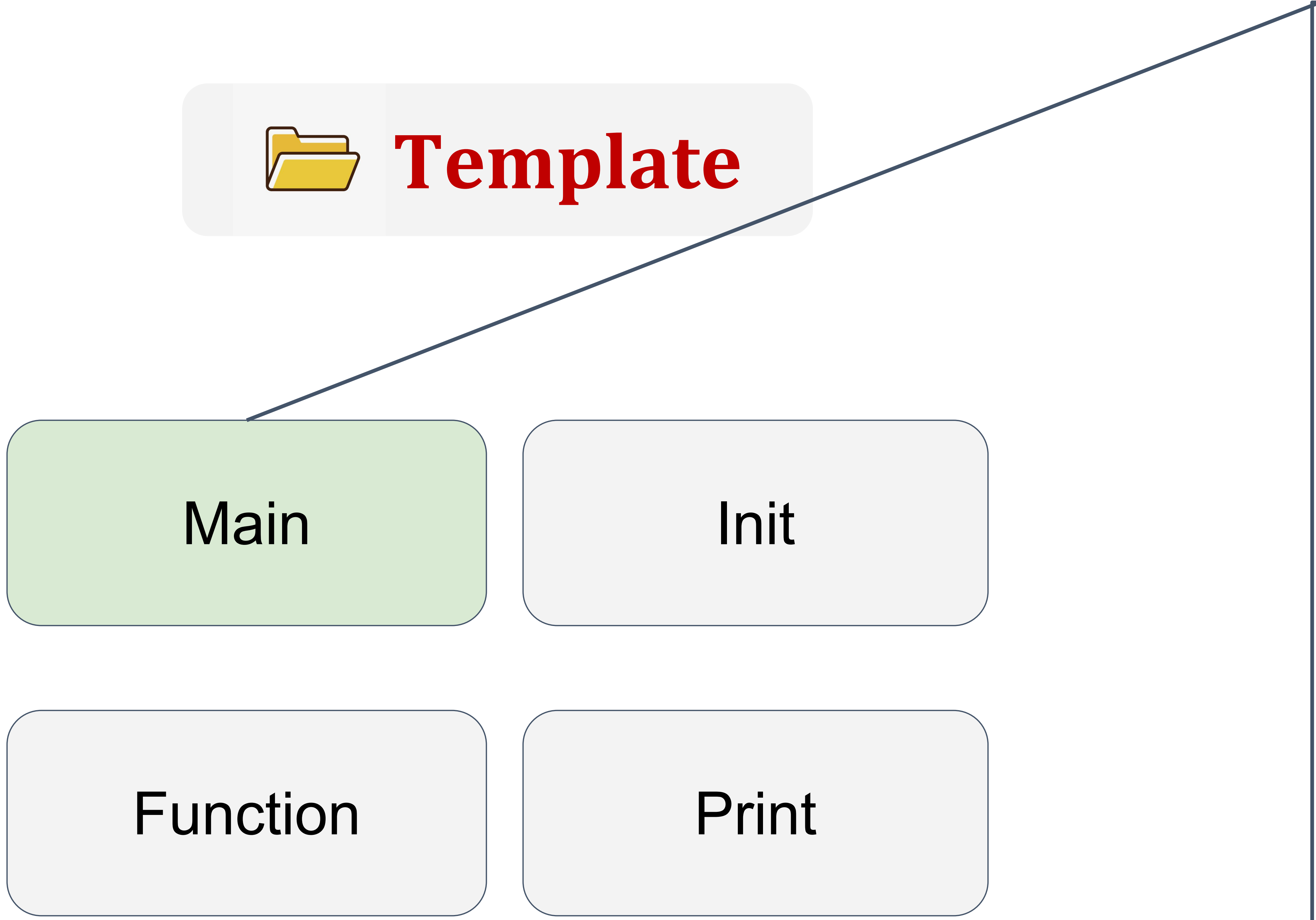
4. C simulation

SW version
vs
HW version

Work Process



1. Template Library

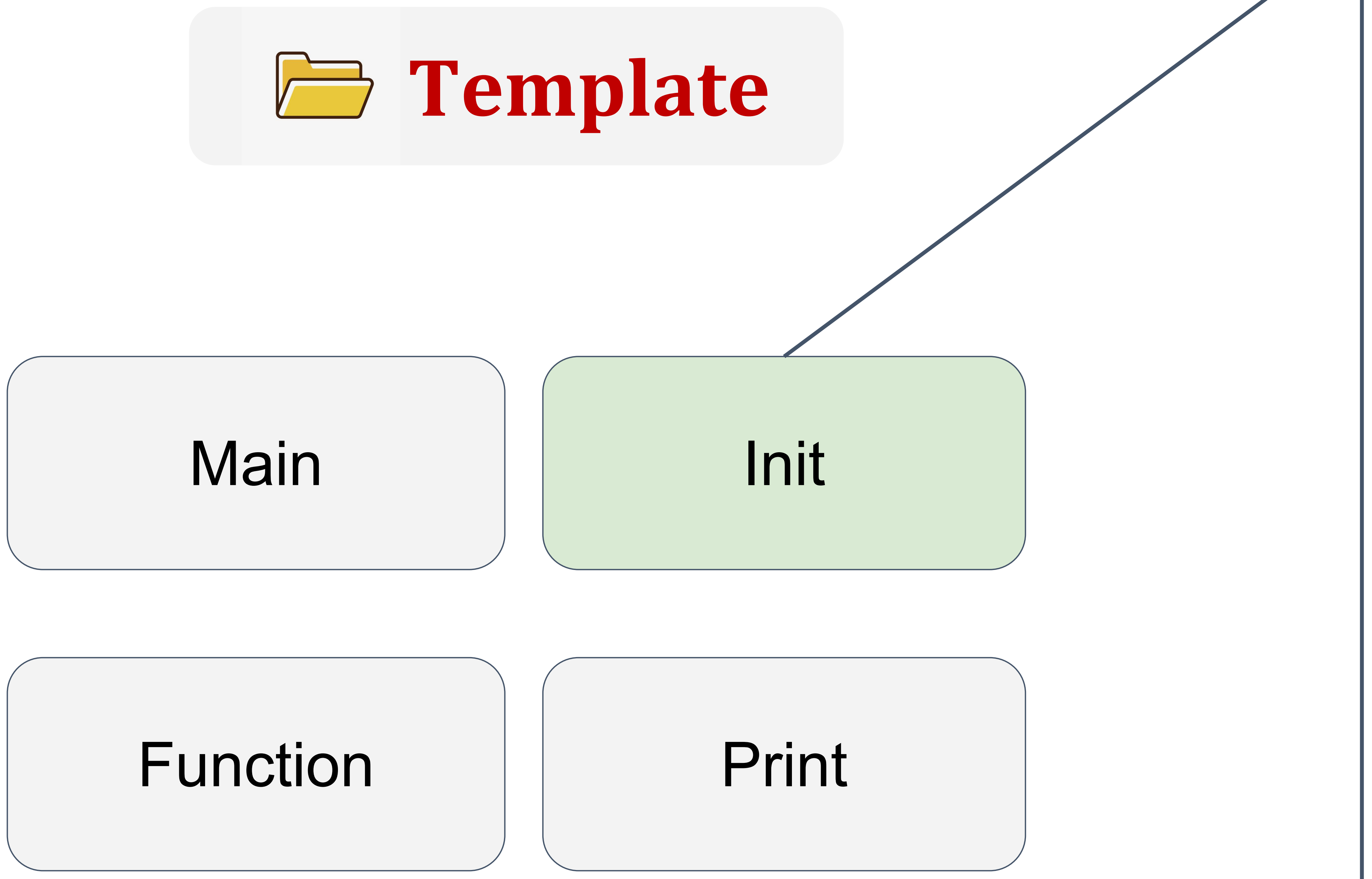


Main

Skeleton code for each program

- C_Verification.txt : For Verifier
- Cpp.txt : For Generator
- Test_cpp.txt : For Generator

1. Template Library

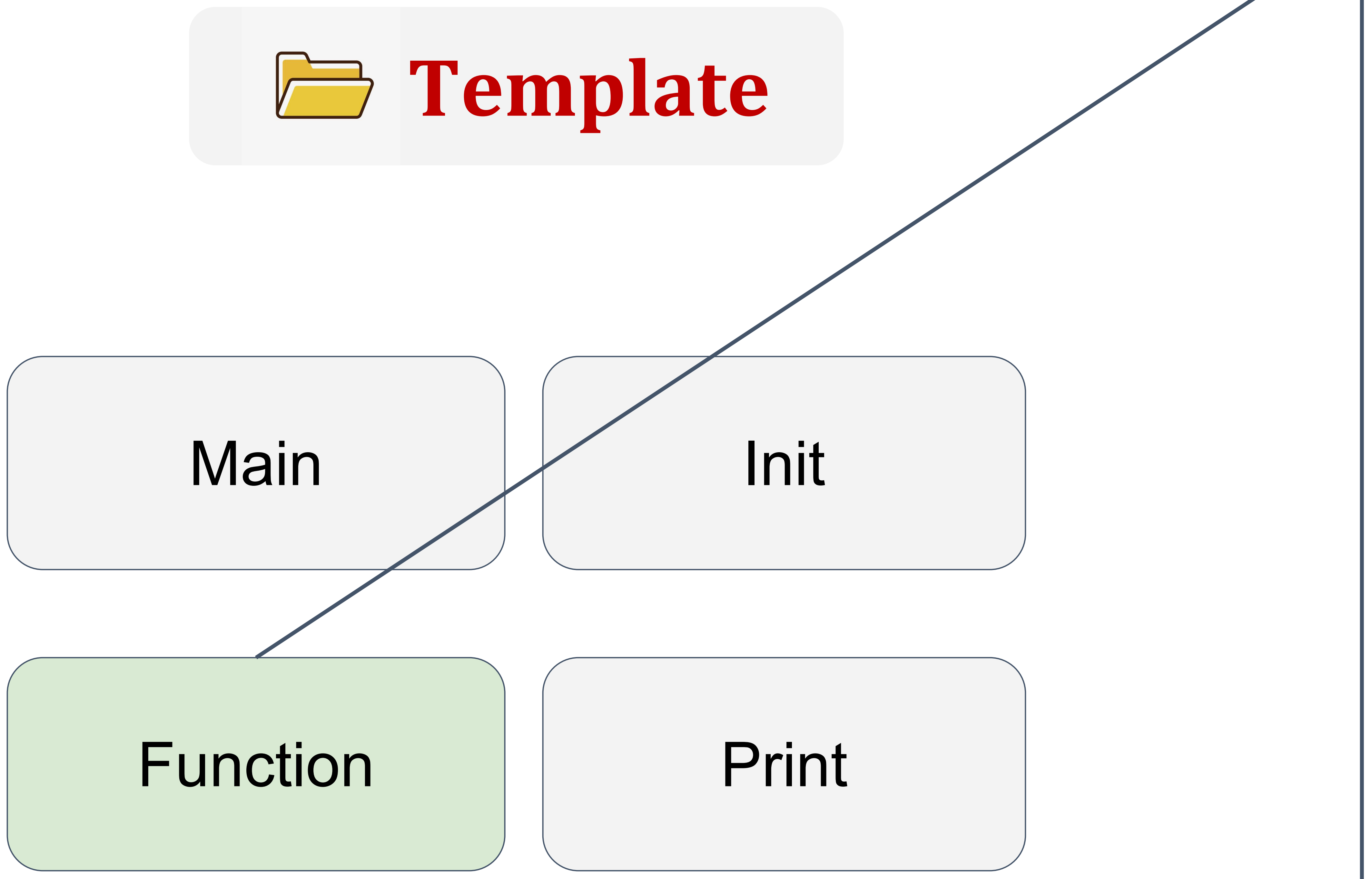


Init

Variable Initialization code

- Init_input.txt : Initialize input
- Init_weight.txt : Initialize weight
- Init_Bias.txt : Initialize bias

1. Template Library



Function

Implemented C functions for CNNs model
Currently applied Model : VGG16, VGG19, Resnet 50

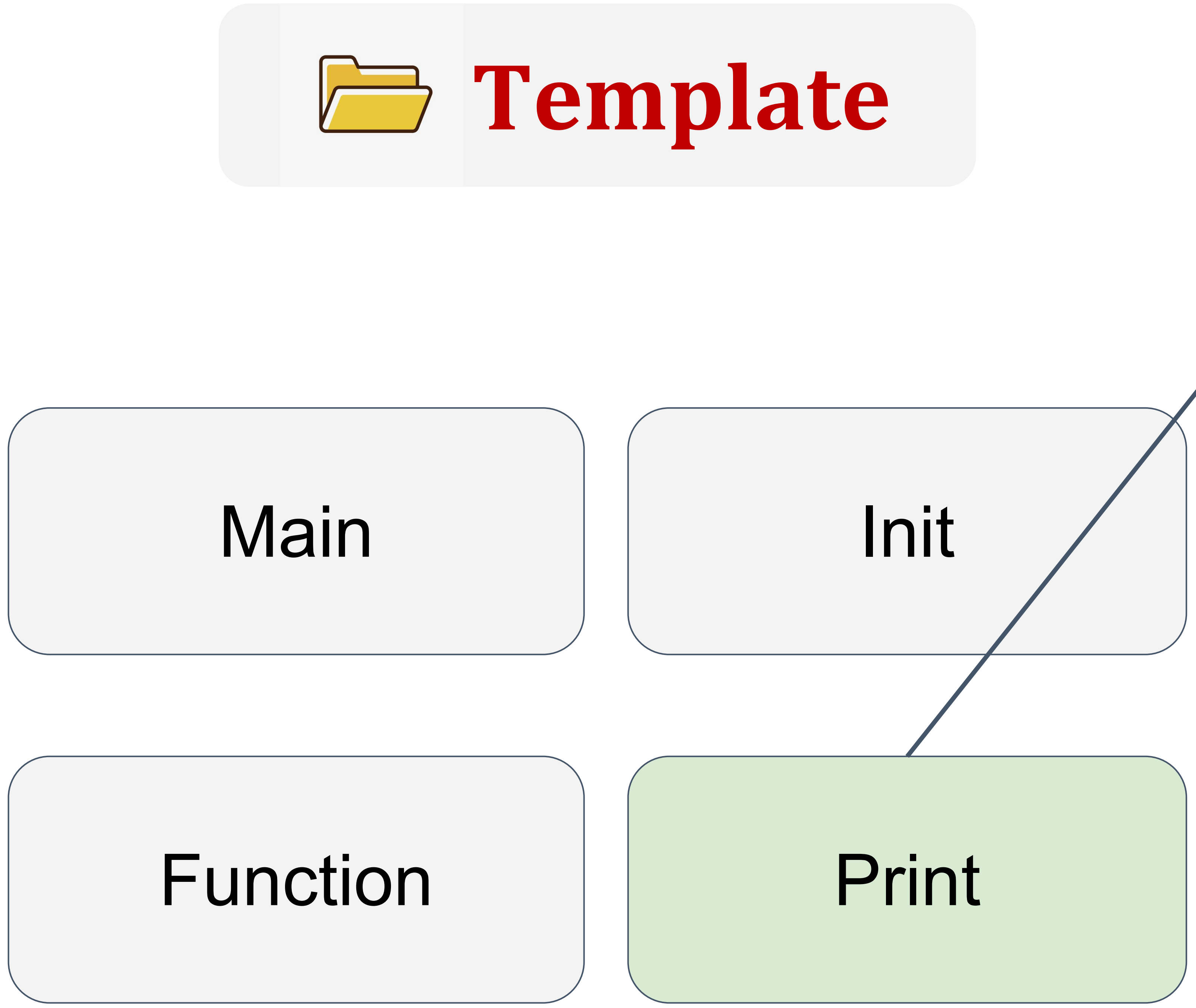
1) Optimized HW function
2) SW function

- MaxPooling2D.txt
- AveragePooling2D.txt
- Convolution2D_Same.txt
- Convolution2D_Valid.txt

……..

- Dense.txt

1. Template Library



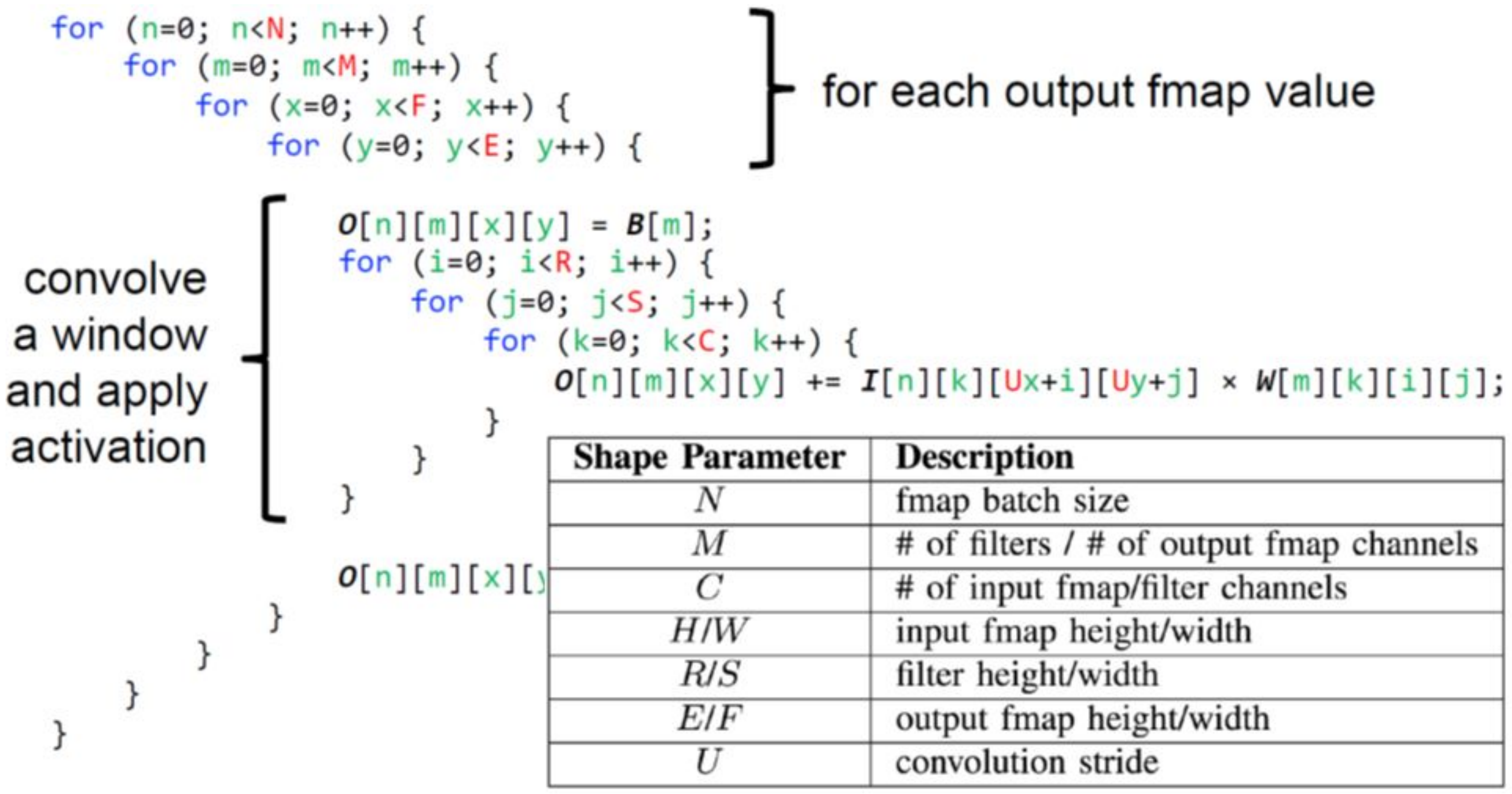
Print

Print function
to compare the results of each layer

- Print4D.txt
- Print3D.txt
- Print2D.txt
- Print1D.txt

How to match keras and C

C


for each output fmap value
convolve a window and apply activation

```
for (n=0; n<N; n++) {
    for (m=0; m<M; m++) {
        for (x=0; x<F; x++) {
            for (y=0; y<E; y++) {
                O[n][m][x][y] = B[m];
                for (i=0; i<R; i++) {
                    for (j=0; j<S; j++) {
                        for (k=0; k<C; k++) {
                            O[n][m][x][y] += I[n][k][Ux+i][Uy+j] × W[m][k][i][j];
                        }
                    }
                }
                O[n][m][x][
            }
        }
    }
}
```

Shape Parameter	Description
N	fmap batch size
M	# of filters / # of output fmap channels
C	# of input fmap/filter channels
H/W	input fmap height/width
R/S	filter height/width
E/F	output fmap height/width
U	convolution stride

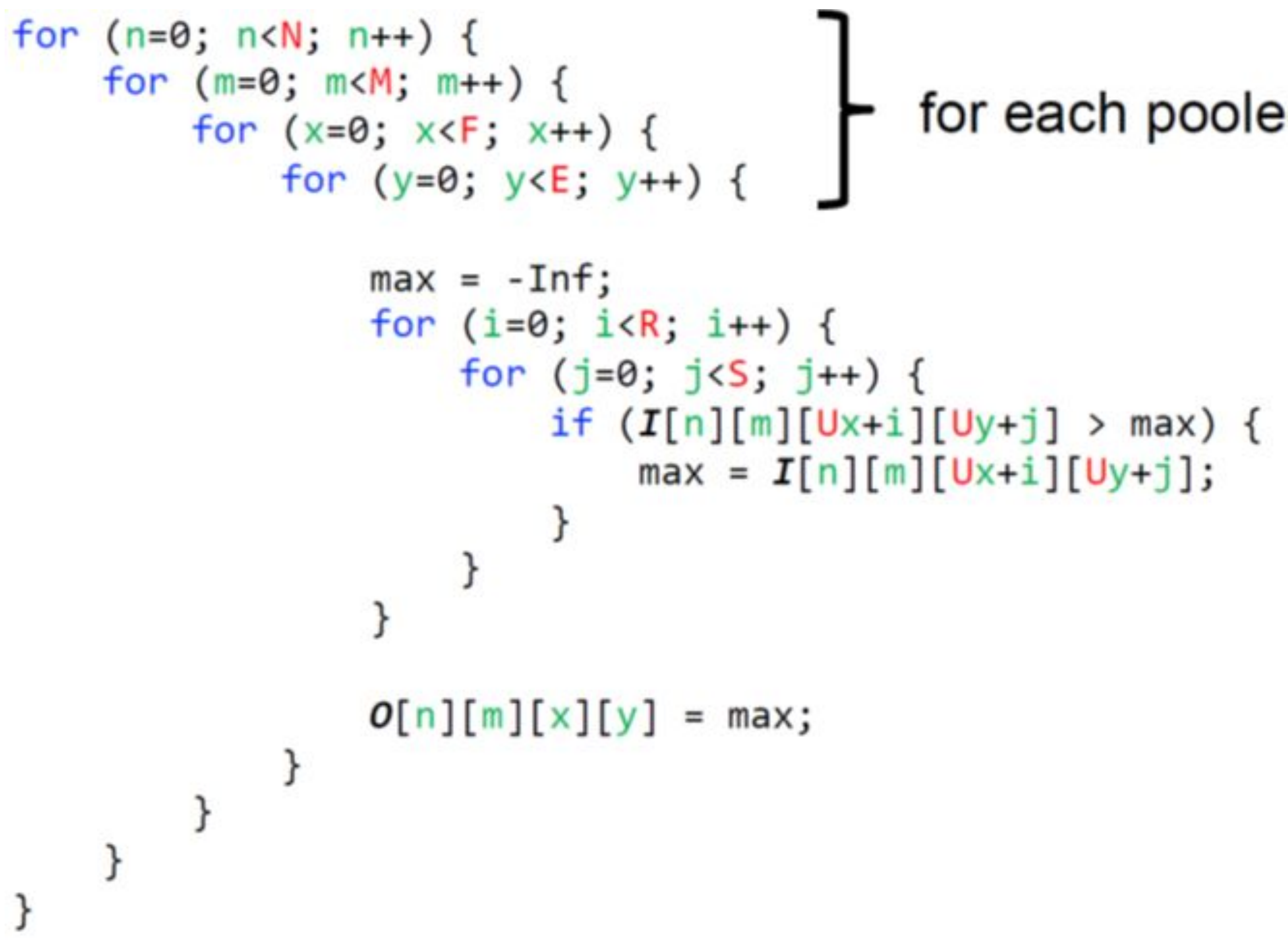
Keras

Conv2D(filters, kernel_size, strides, padding, data_format, dilation_rate, activation, use_bias , kernel_initializer, bias_initializer, kernel_regularizer, bias_regularizer, activity_regularizer, kernel_constraint, bias_constraint)

- filters : number of output filter
- kernel_size : filter size
- padding : ‘same’ , ‘valid’
- data_format : ‘channels_last’ , ‘channel_first’
- dilation_rate : k = 1, 2, 4 ..
- activation : activation function
- use bias : W*x + bias (True/ False)
- kernel_initializer : weight initializer
- ...

How to match keras and C

C



```
for (n=0; n<N; n++) {
    for (m=0; m<M; m++) {
        for (x=0; x<F; x++) {
            for (y=0; y<E; y++) {

                max = -Inf;
                for (i=0; i<R; i++) {
                    for (j=0; j<S; j++) {
                        if (I[n][m][Ux+i][Uy+j] > max) {
                            max = I[n][m][Ux+i][Uy+j];
                        }
                    }
                }

                O[n][m][x][y] = max;
            }
        }
    }
}
```

for each pooled value

find the max
with in a window

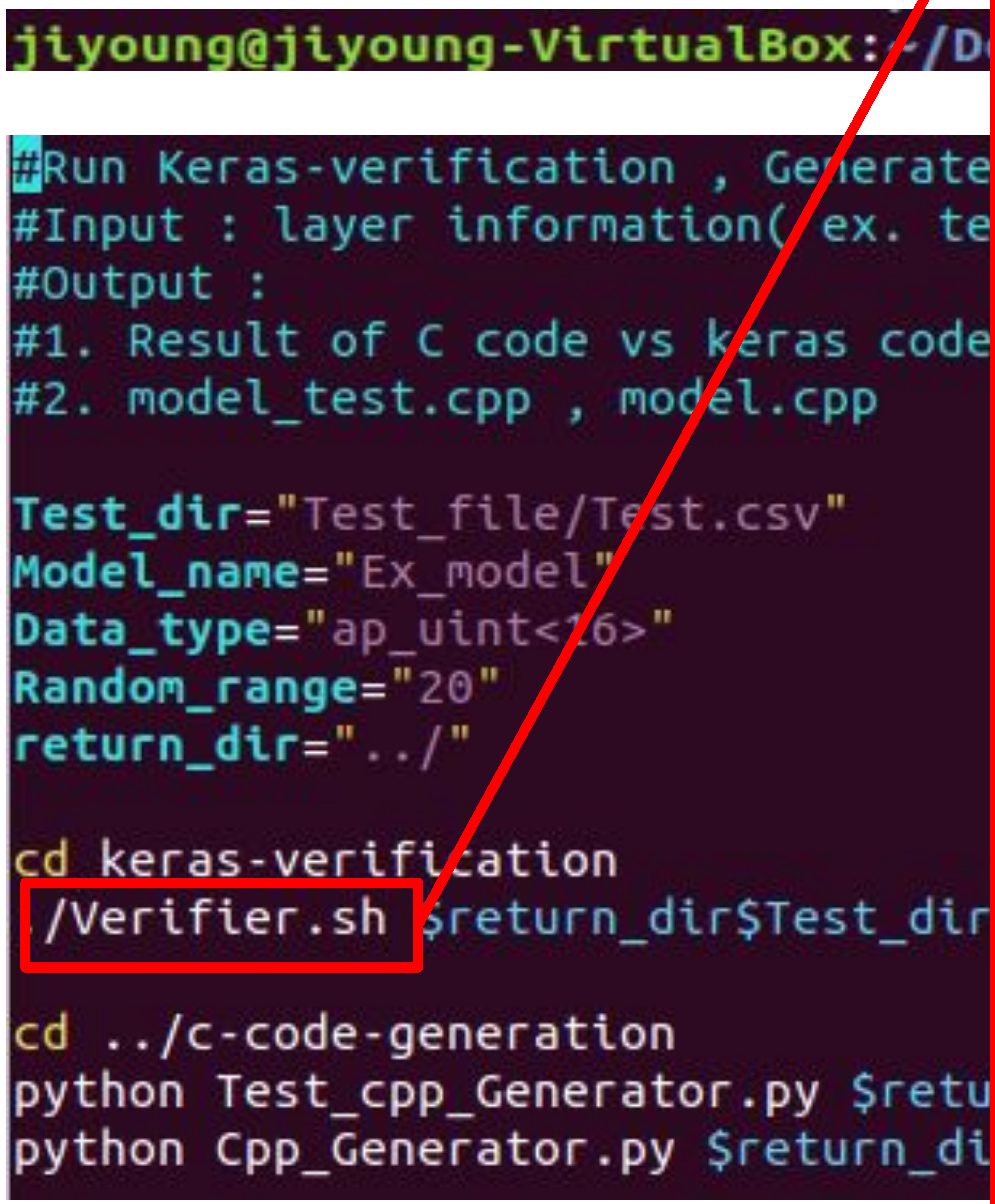
Keras

MaxPooling2D(pool-size , strides , padding, data_format)

- pool-size : pooling window
- strides : step
- padding : ‘same’ (Add zero-padding)
  ‘valid’ (Traverse only with in the image)
- data_format : ‘channels_last’ (batch, channel, width, height)
  ‘channels_first’ (batch, width, height, channel)

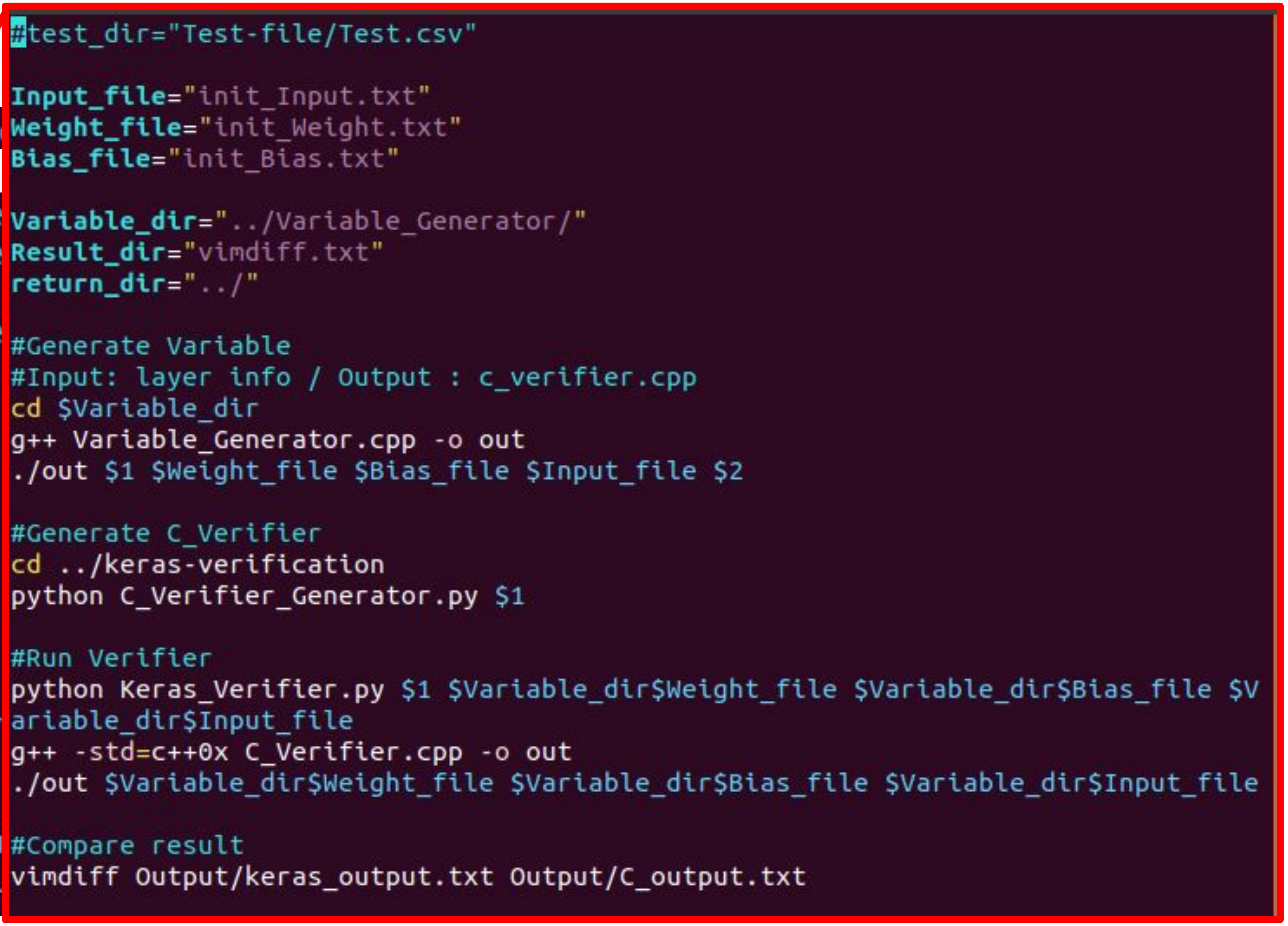
2) Result

1. Run



```
jiyoung@jiyoung-VirtualBox:~/D
```

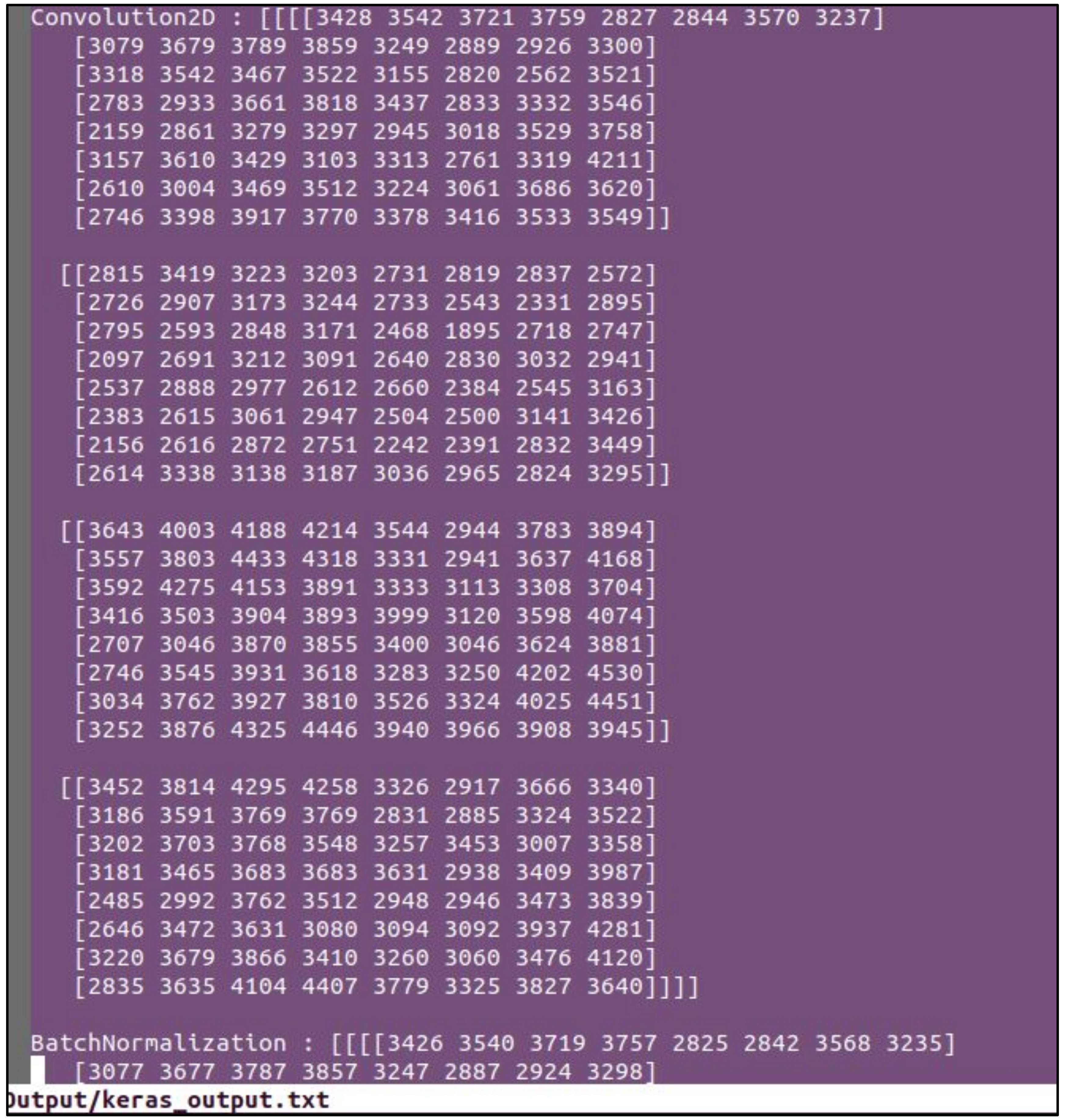
```
#Run Keras-verification , Generate
#Input : layer information( ex. te
#Output :
#1. Result of C code vs keras code
#2. model_test.cpp , model.cpp

Test_dir="Test_file/Test.csv"
Model_name="Ex_model"
Data_type="ap_uint<16>"
Random_range="20"
return_dir="../"

cd keras-verification
./Verifier.sh $return_dir$Test_dir

cd ../c-code-generation
python Test_cpp_Generator.py $retu
python Cpp_Generator.py $return_di
```



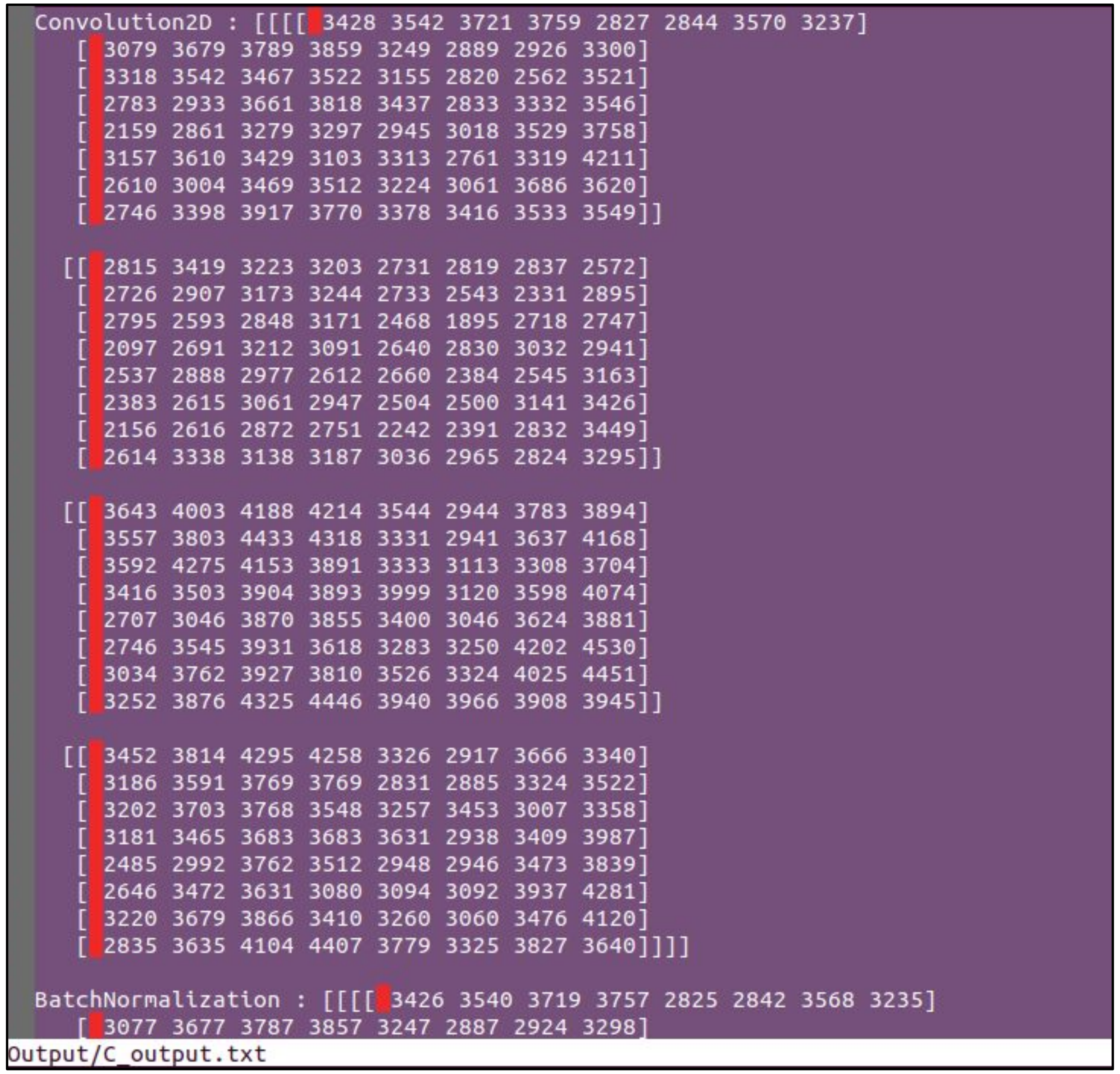
```
#test_dir="Test-file/Test.csv"

Input_file="init_Input.txt"
Weight_file="init_Weight.txt"
Bias_file="init_Bias.txt"

Variable_dir="../Variable_Generator/"
Result_dir="vimdiff.txt"
return_dir="../"

#Generate Variable
#Input: layer info / Output : c_verifier.cpp
cd $Variable_dir
g++ Variable_Generator.cpp -o out
./out $1 $Weight_file $Bias_file $Input_file $2

#Generate C_Verifier
cd ../keras-verification
python C_Verifier_Generator.py $1

#Run Verifier
python Keras_Verifier.py $1 $Variable_dir$Weight_file $Variable_dir$Bias_file $V
ariable_dir$Input_file
g++ -std=c++0x C_Verifier.cpp -o out
./out $Variable_dir$Weight_file $Variable_dir$Bias_file $Variable_dir$Input_file

#Compare result
vimdiff Output/keras_output.txt Output/C_output.txt
```

2) Result

2. Print Diff



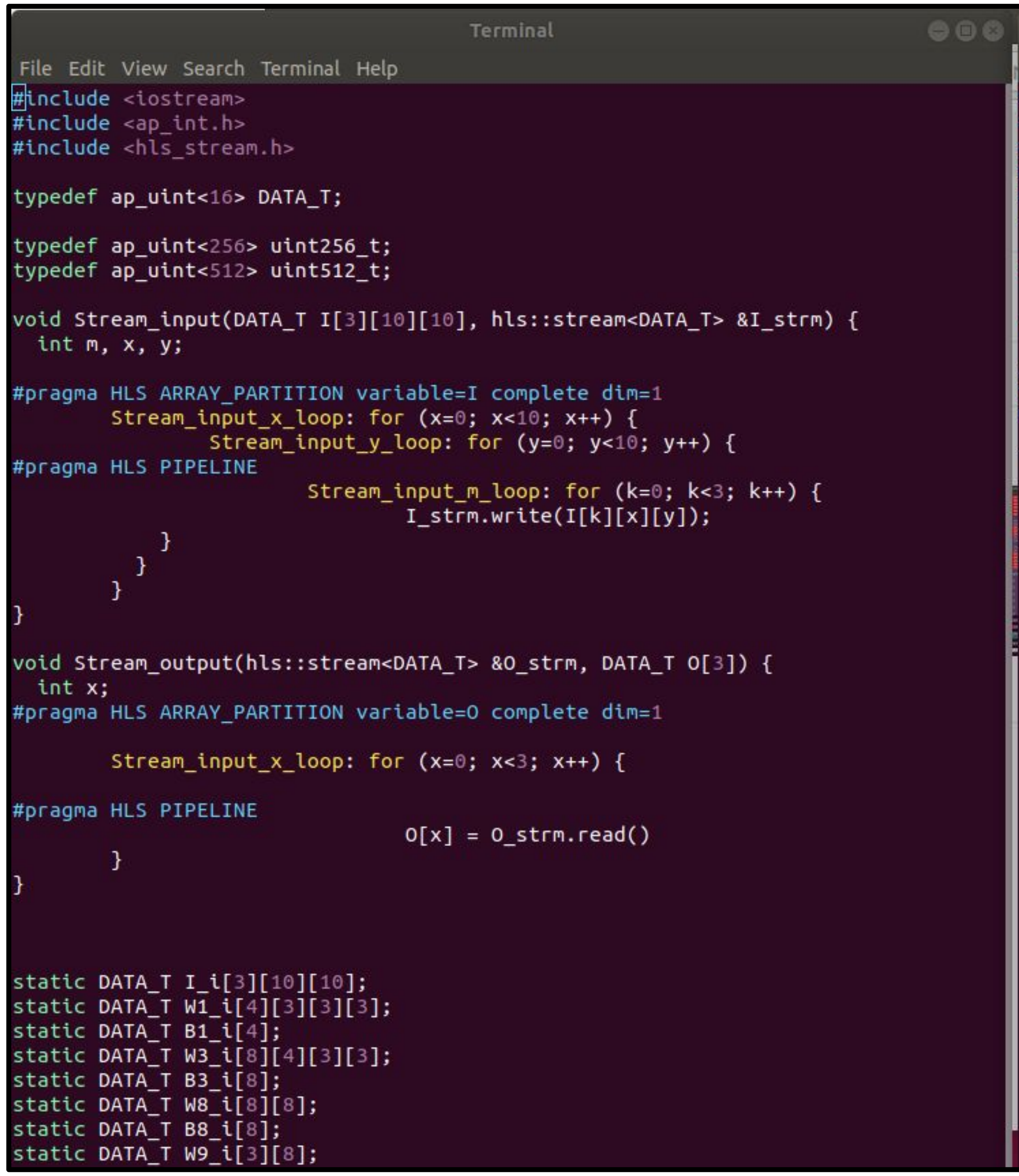
```
Convolution2D : [[[[3428 3542 3721 3759 2827 2844 3570 3237]
   [3079 3679 3789 3859 3249 2889 2926 3300]
   [3318 3542 3467 3522 3155 2820 2562 3521]
   [2783 2933 3661 3818 3437 2833 3332 3546]
   [2159 2861 3279 3297 2945 3018 3529 3758]
   [3157 3610 3429 3103 3313 2761 3319 4211]
   [2610 3004 3469 3512 3224 3061 3686 3620]
   [2746 3398 3917 3770 3378 3416 3533 3549]]

  [[2815 3419 3223 3203 2731 2819 2837 2572]
   [2726 2907 3173 3244 2733 2543 2331 2895]
   [2795 2593 2848 3171 2468 1895 2718 2747]
   [2097 2691 3212 3091 2640 2830 3032 2941]
   [2537 2888 2977 2612 2660 2384 2545 3163]
   [2383 2615 3061 2947 2504 2500 3141 3426]
   [2156 2616 2872 2751 2242 2391 2832 3449]
   [2614 3338 3138 3187 3036 2965 2824 3295]]

  [[3643 4003 4188 4214 3544 2944 3783 3894]
   [3557 3803 4433 4318 3331 2941 3637 4168]
   [3592 4275 4153 3891 3333 3113 3308 3704]
   [3416 3503 3904 3893 3999 3120 3598 4074]
   [2707 3046 3870 3855 3400 3046 3624 3881]
   [2746 3545 3931 3618 3283 3250 4202 4530]
   [3034 3762 3927 3810 3526 3324 4025 4451]
   [3252 3876 4325 4446 3940 3966 3908 3945]]

  [[3452 3814 4295 4258 3326 2917 3666 3340]
   [3186 3591 3769 3769 2831 2885 3324 3522]
   [3202 3703 3768 3548 3257 3453 3007 3358]
   [3181 3465 3683 3683 3631 2938 3409 3987]
   [2485 2992 3762 3512 2948 2946 3473 3839]
   [2646 3472 3631 3080 3094 3092 3937 4281]
   [3220 3679 3866 3410 3260 3060 3476 4120]
   [2835 3635 4104 4407 3779 3325 3827 3640]]]]

BatchNormalization : [[[[3426 3540 3719 3757 2825 2842 3568 3235]
   [3077 3677 3787 3857 3247 2887 2924 3298]
Output/keras_output.txt
```



```
Convolution2D : [[[[ 3428 3542 3721 3759 2827 2844 3570 3237]
   [ 3079 3679 3789 3859 3249 2889 2926 3300]
   [ 3318 3542 3467 3522 3155 2820 2562 3521]
   [ 2783 2933 3661 3818 3437 2833 3332 3546]
   [ 2159 2861 3279 3297 2945 3018 3529 3758]
   [ 3157 3610 3429 3103 3313 2761 3319 4211]
   [ 2610 3004 3469 3512 3224 3061 3686 3620]
   [ 2746 3398 3917 3770 3378 3416 3533 3549]]

  [[ 2815 3419 3223 3203 2731 2819 2837 2572]
   [ 2726 2907 3173 3244 2733 2543 2331 2895]
   [ 2795 2593 2848 3171 2468 1895 2718 2747]
   [ 2097 2691 3212 3091 2640 2830 3032 2941]
   [ 2537 2888 2977 2612 2660 2384 2545 3163]
   [ 2383 2615 3061 2947 2504 2500 3141 3426]
   [ 2156 2616 2872 2751 2242 2391 2832 3449]
   [ 2614 3338 3138 3187 3036 2965 2824 3295]]

  [[ 3643 4003 4188 4214 3544 2944 3783 3894]
   [ 3557 3803 4433 4318 3331 2941 3637 4168]
   [ 3592 4275 4153 3891 3333 3113 3308 3704]
   [ 3416 3503 3904 3893 3999 3120 3598 4074]
   [ 2707 3046 3870 3855 3400 3046 3624 3881]
   [ 2746 3545 3931 3618 3283 3250 4202 4530]
   [ 3034 3762 3927 3810 3526 3324 4025 4451]
   [ 3252 3876 4325 4446 3940 3966 3908 3945]]

  [[ 3452 3814 4295 4258 3326 2917 3666 3340]
   [ 3186 3591 3769 3769 2831 2885 3324 3522]
   [ 3202 3703 3768 3548 3257 3453 3007 3358]
   [ 3181 3465 3683 3683 3631 2938 3409 3987]
   [ 2485 2992 3762 3512 2948 2946 3473 3839]
   [ 2646 3472 3631 3080 3094 3092 3937 4281]
   [ 3220 3679 3866 3410 3260 3060 3476 4120]
   [ 2835 3635 4104 4407 3779 3325 3827 3640]]]]

BatchNormalization : [[[[ 3426 3540 3719 3757 2825 2842 3568 3235]
   [ 3077 3677 3787 3857 3247 2887 2924 3298]
Output/C_output.txt
```

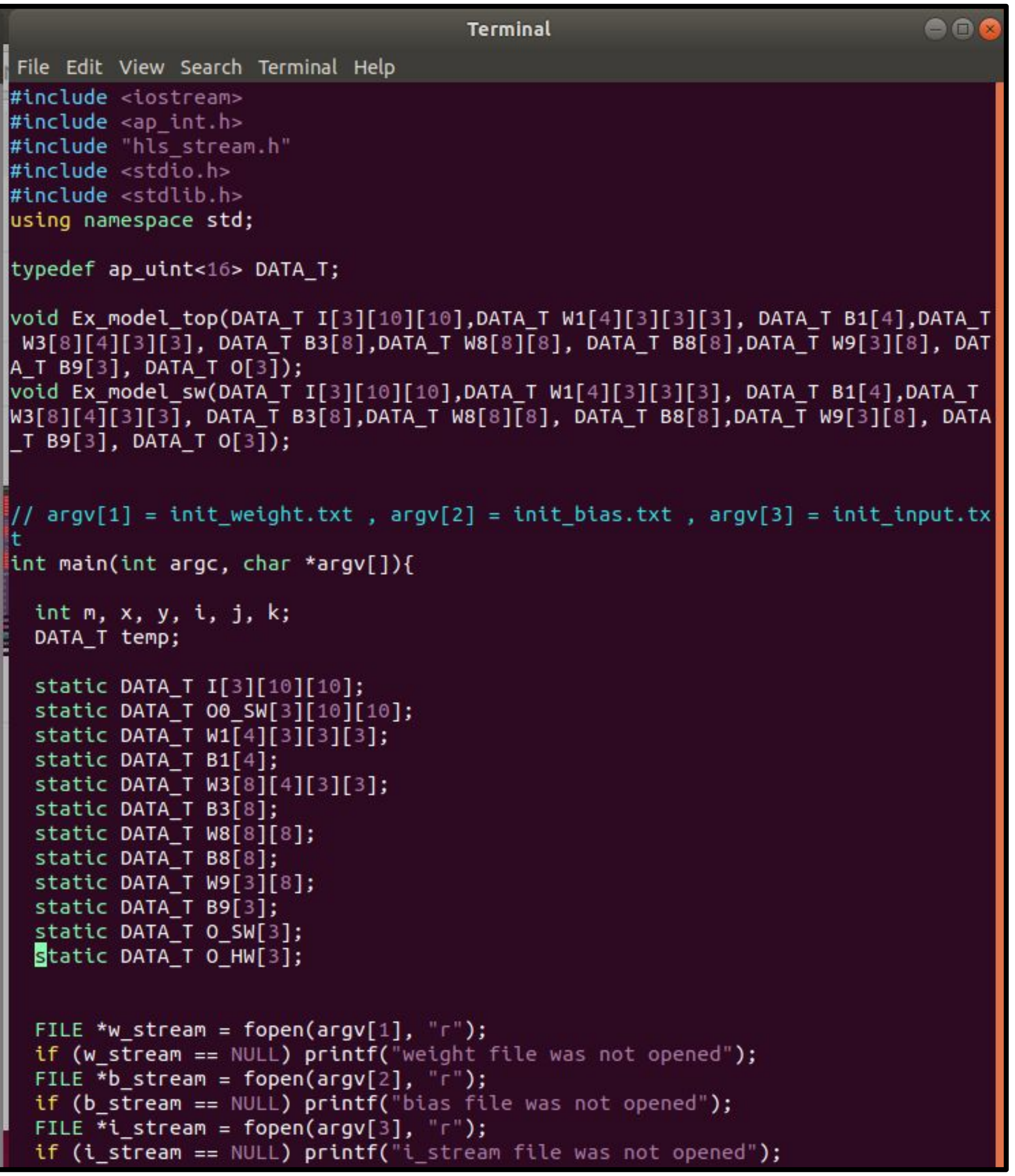
2) Result

3. Generate File



```
#include <iostream>
#include <ap_int.h>
#include <hls_stream.h>

typedef ap_uint<16> DATA_T;

typedef ap_uint<256> uint256_t;
typedef ap_uint<512> uint512_t;

void Stream_input(DATA_T I[3][10][10], hls::stream<DATA_T> &I_strm) {
  int m, x, y;

#pragma HLS ARRAY_PARTITION variable=I complete dim=1
        Stream_input_x_loop: for (x=0; x<10; x++) {
                Stream_input_y_loop: for (y=0; y<10; y++) {
#pragma HLS PIPELINE
                        Stream_input_m_loop: for (k=0; k<3; k++) {
                                I_strm.write(I[k][x][y]);
                }
            }
        }
}

void Stream_output(hls::stream<DATA_T> &O_strm, DATA_T O[3]) {
  int x;
#pragma HLS ARRAY_PARTITION variable=O complete dim=1

        Stream_input_x_loop: for (x=0; x<3; x++) {

#pragma HLS PIPELINE
                                O[x] = O_strm.read()
        }
}



static DATA_T I_i[3][10][10];
static DATA_T W1_i[4][3][3][3];
static DATA_T B1_i[4];
static DATA_T W3_i[8][4][3][3];
static DATA_T B3_i[8];
static DATA_T W8_i[8][8];
static DATA_T B8_i[8];
static DATA_T W9_i[3][8];
```

< Model.cpp >



```
#include <iostream>
#include <ap_int.h>
#include "hls_stream.h"
#include <stdio.h>
#include <stdlib.h>
using namespace std;

typedef ap_uint<16> DATA_T;

void Ex_model_top(DATA_T I[3][10][10],DATA_T W1[4][3][3][3], DATA_T B1[4],DATA_T
 W3[8][4][3][3], DATA_T B3[8],DATA_T W8[8][8], DATA_T B8[8],DATA_T W9[3][8], DAT
A_T B9[3], DATA_T O[3]);
void Ex_model_sw(DATA_T I[3][10][10],DATA_T W1[4][3][3][3], DATA_T B1[4],DATA_T
W3[8][4][3][3], DATA_T B3[8],DATA_T W8[8][8], DATA_T B8[8],DATA_T W9[3][8], DATA
_T B9[3], DATA_T O[3]);


// argv[1] = init_weight.txt , argv[2] = init_bias.txt , argv[3] = init_input.tx
t
int main(int argc, char *argv[]){

  int m, x, y, i, j, k;
  DATA_T temp;

  static DATA_T I[3][10][10];
  static DATA_T O0_SW[3][10][10];
  static DATA_T W1[4][3][3][3];
  static DATA_T B1[4];
  static DATA_T W3[8][4][3][3];
  static DATA_T B3[8];
  static DATA_T W8[8][8];
  static DATA_T B8[8];
  static DATA_T W9[3][8];
  static DATA_T B9[3];
  static DATA_T O_SW[3];
  static DATA_T O_HW[3];


  FILE *w_stream = fopen(argv[1], "r");
  if (w_stream == NULL) printf("weight file was not opened");
  FILE *b_stream = fopen(argv[2], "r");
  if (b_stream == NULL) printf("bias file was not opened");
  FILE *i_stream = fopen(argv[3], "r");
  if (i_stream == NULL) printf("i_stream file was not opened");
```

< Testbench.cpp >

Conclusion

Current State

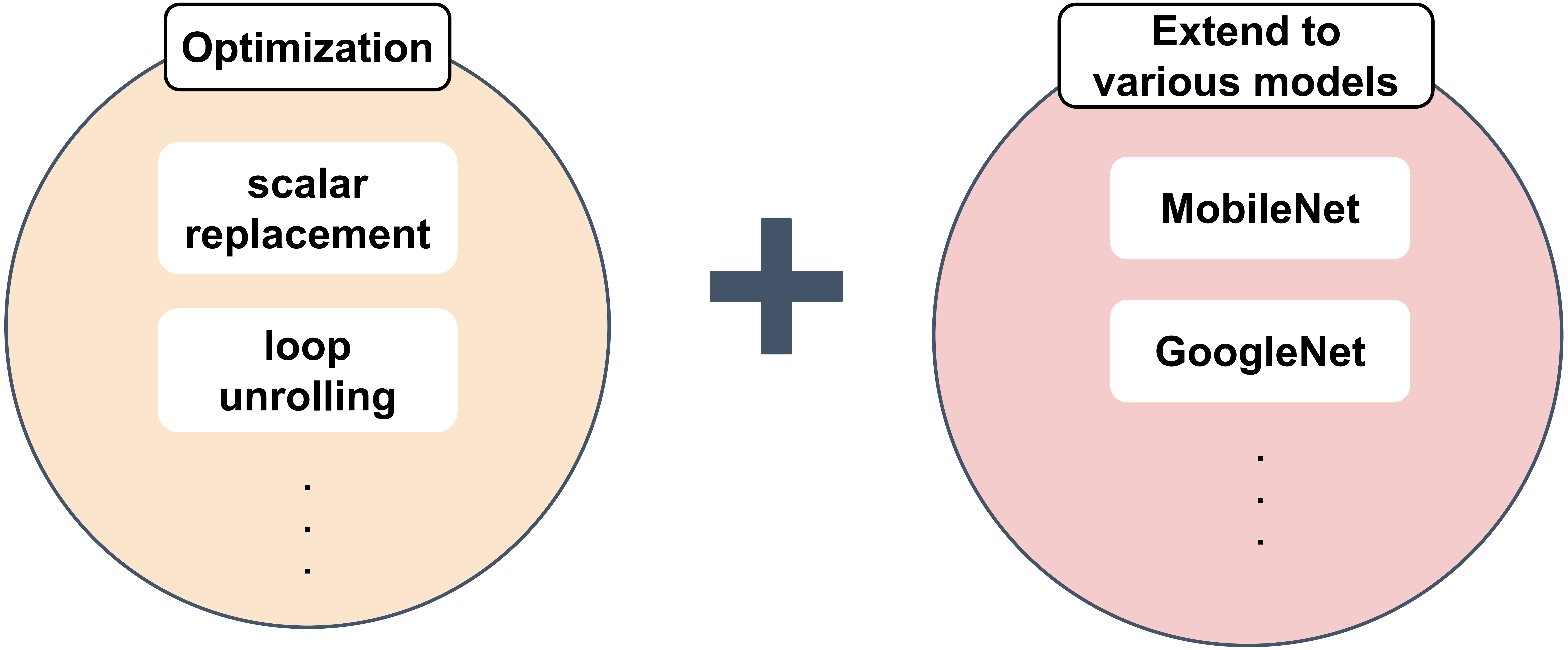
Xiao, Qingcheng, et al. "Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on FPGAs." *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017.

Reduce memory size

Use new optimizing ways

Minimize computational latency

Future Plan


Optimization
scalar replacement
loop unrolling
.
.
.
Extend to various models
MobileNet
GoogleNet
.
.
.

THANK YOU