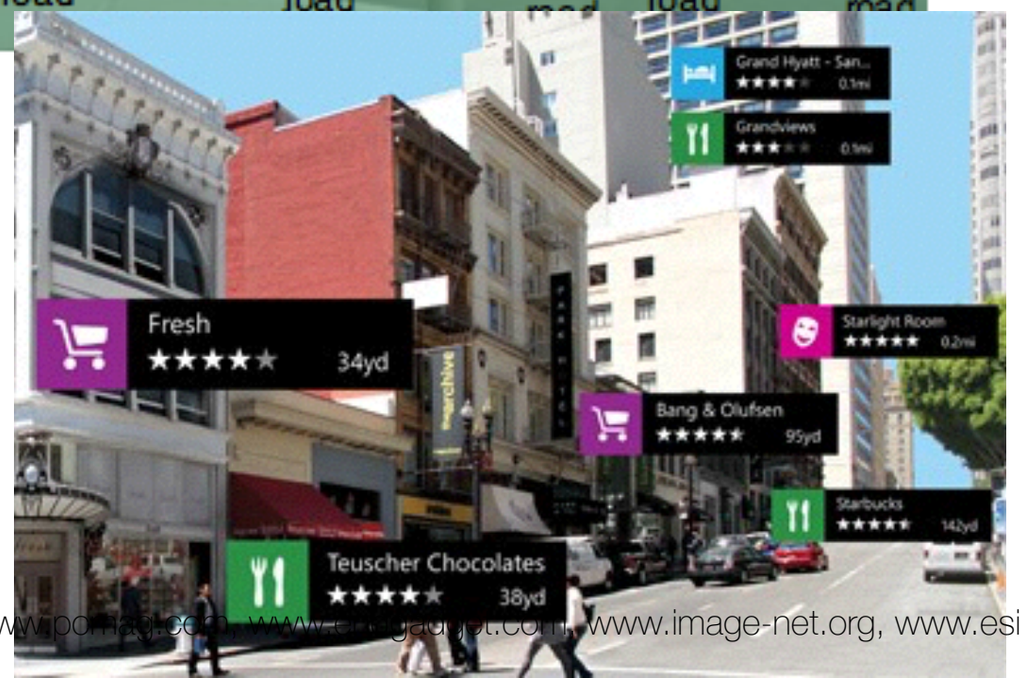
Accelerating

Convolutional Neural Networks

on a
Field-Programmable Gate Array

Idea and Mission

- Object Recognition / Scene Labeling



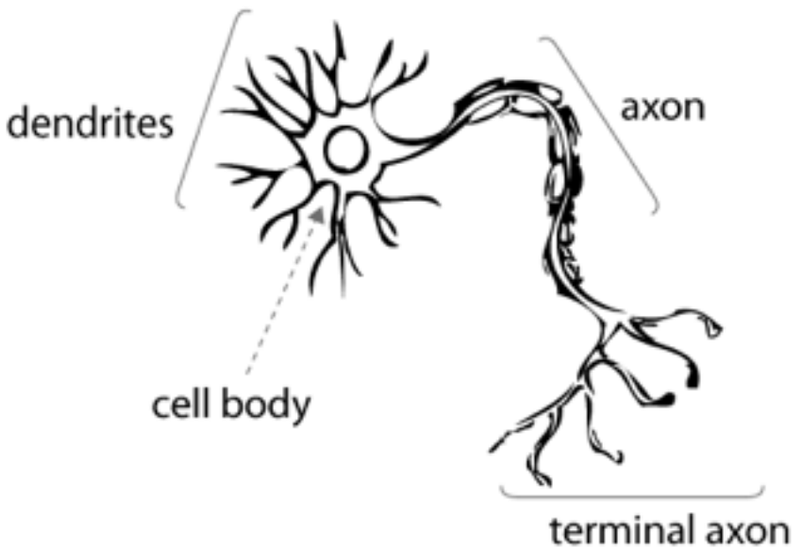
Neural Networks

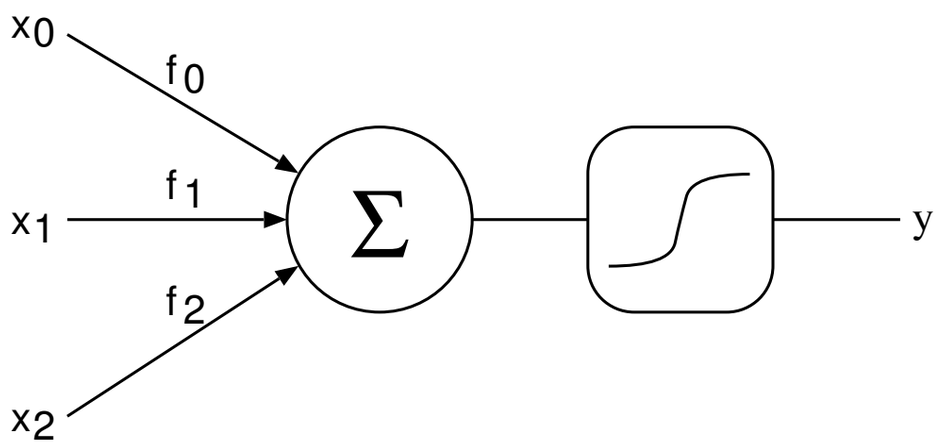
- Computation Algorithms inspired by Nervous System
- Artificial Neurons map many low-complexity inputs

↓

one output

with higher abstraction level

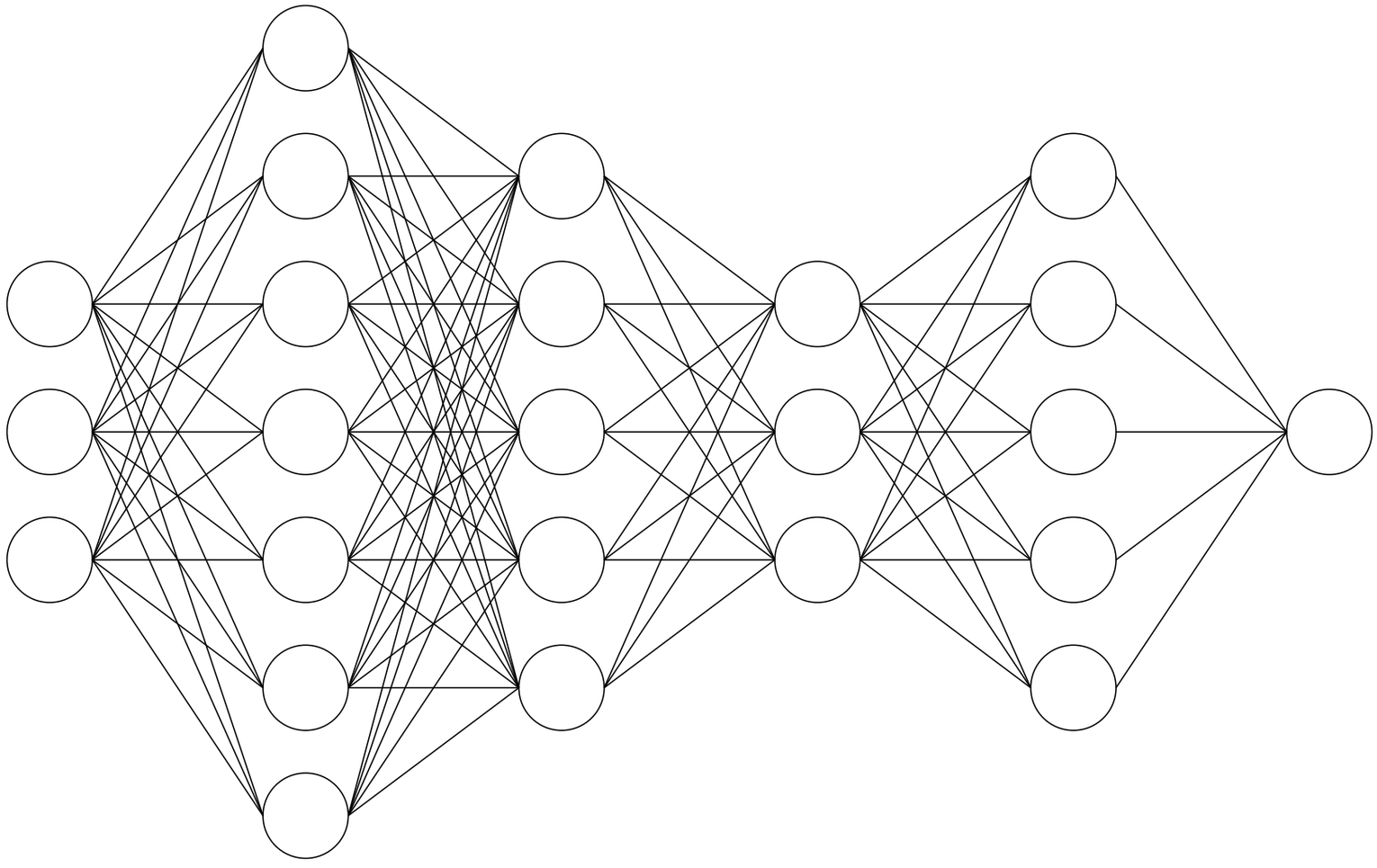




Images: www.biologycorner.com

Neural Networks

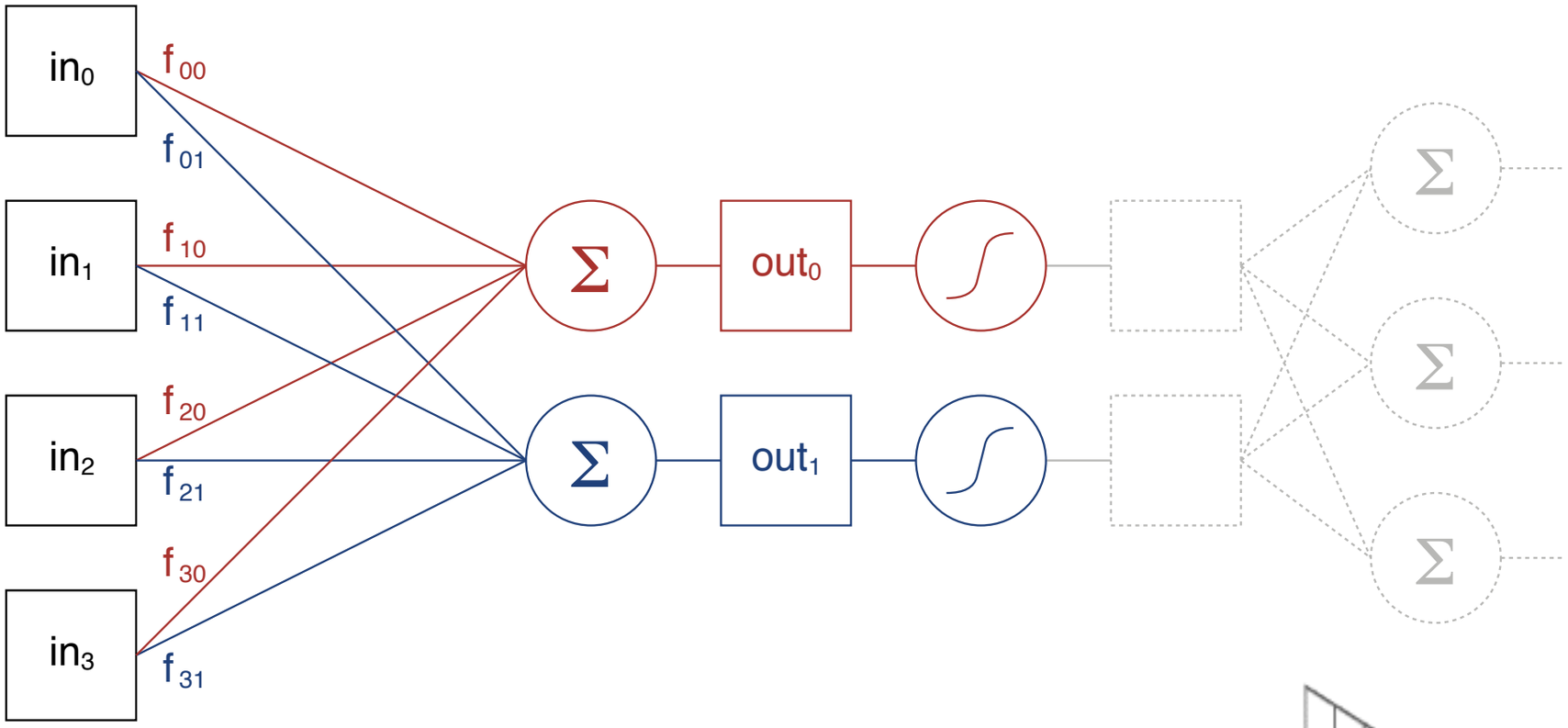
- Many Neurons form a **Neural Network**



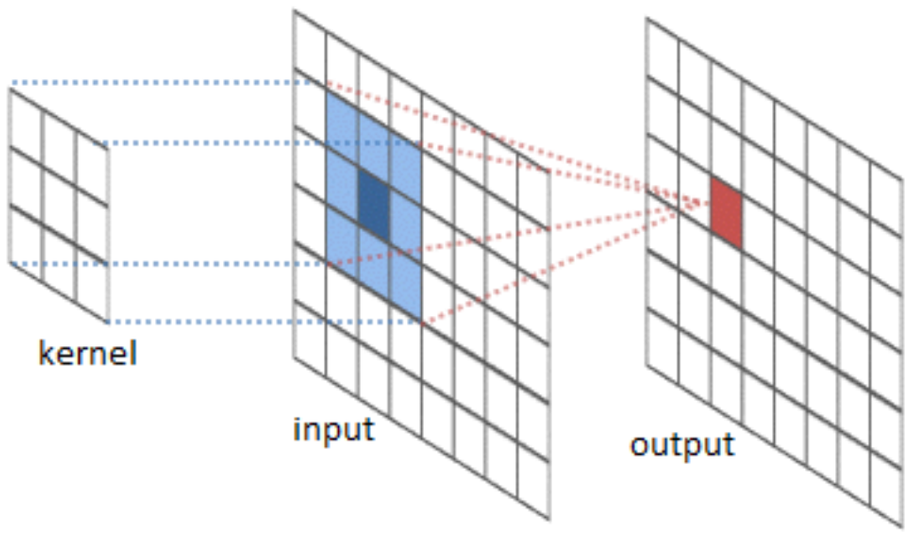
Huge amount of coefficients → Training of Neural Network.

Optimization for “best approximation of desired output” with known data.

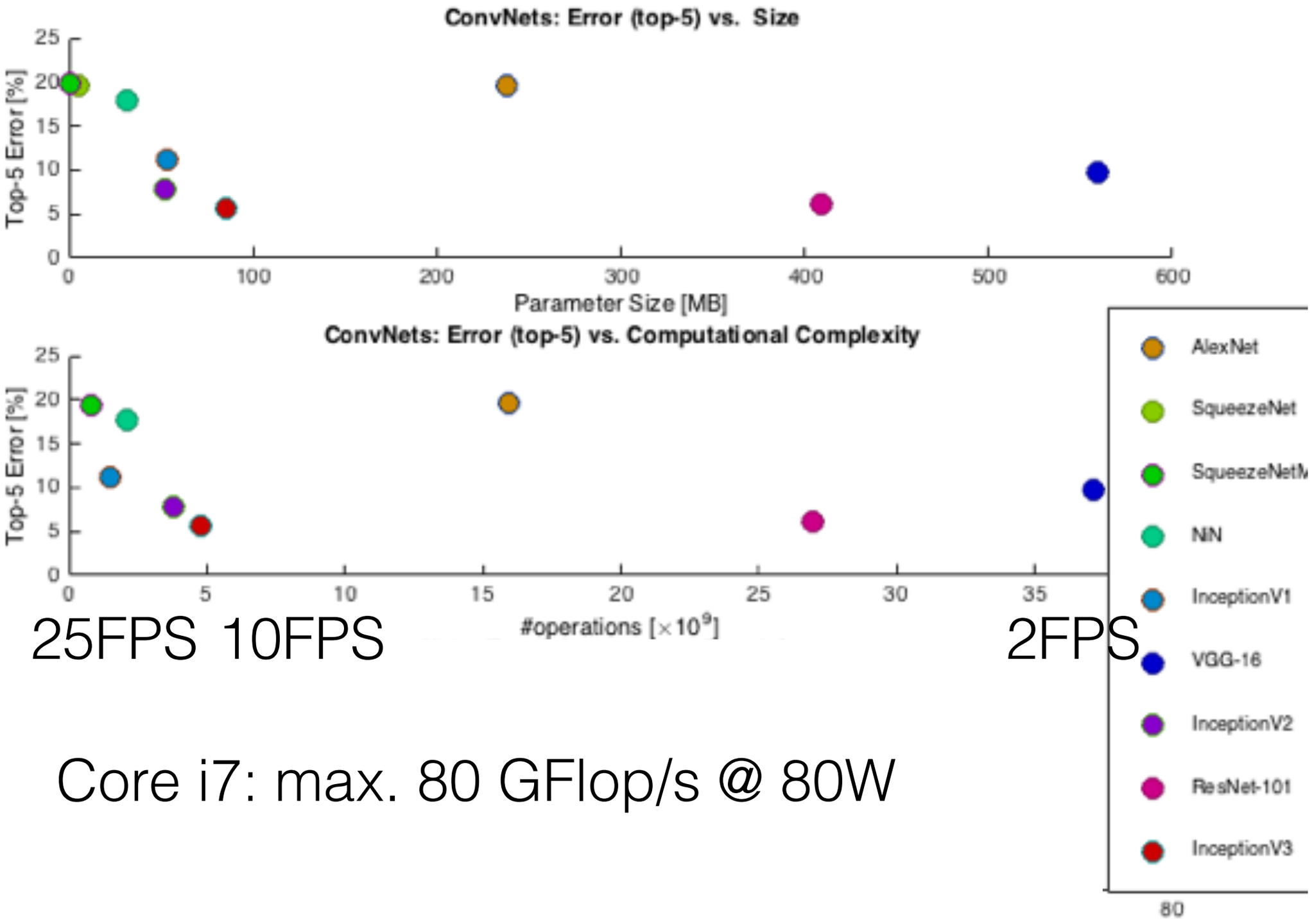
Convolutional Neural Networks



- Convolutional Neural Networks
  - Operate on 2D data (e.g. images)
  - Perform “weighting” by filter kernels
  - Capture neighborhood relations between pixels



Accuracy vs. Complexity of CNNs

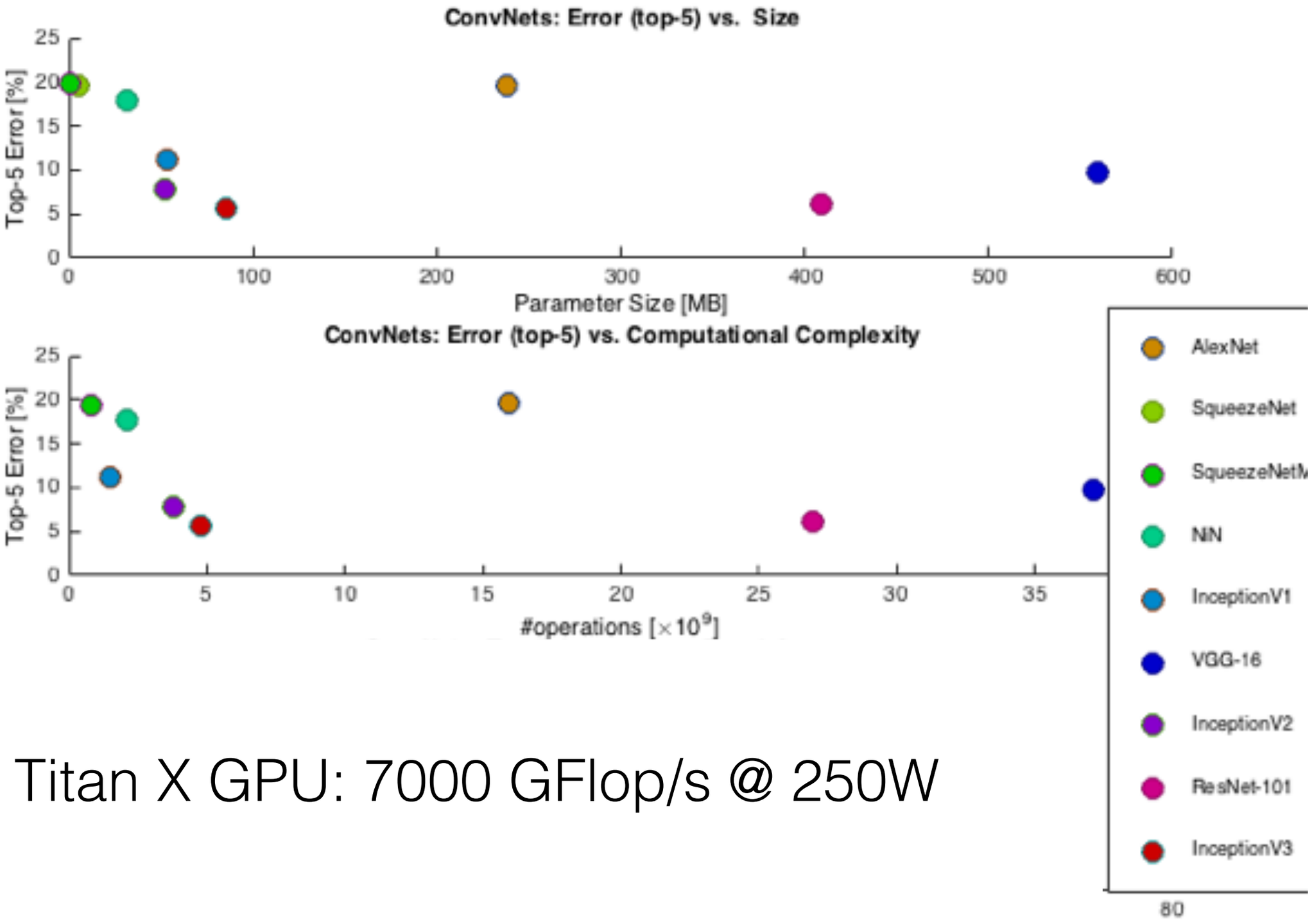


25FPS 10FPS 2FPS

Core i7: max. 80 GFlop/s @ 80W

https://de.wikipedia.org/wiki/Floating_Point_Operations_Per_Second

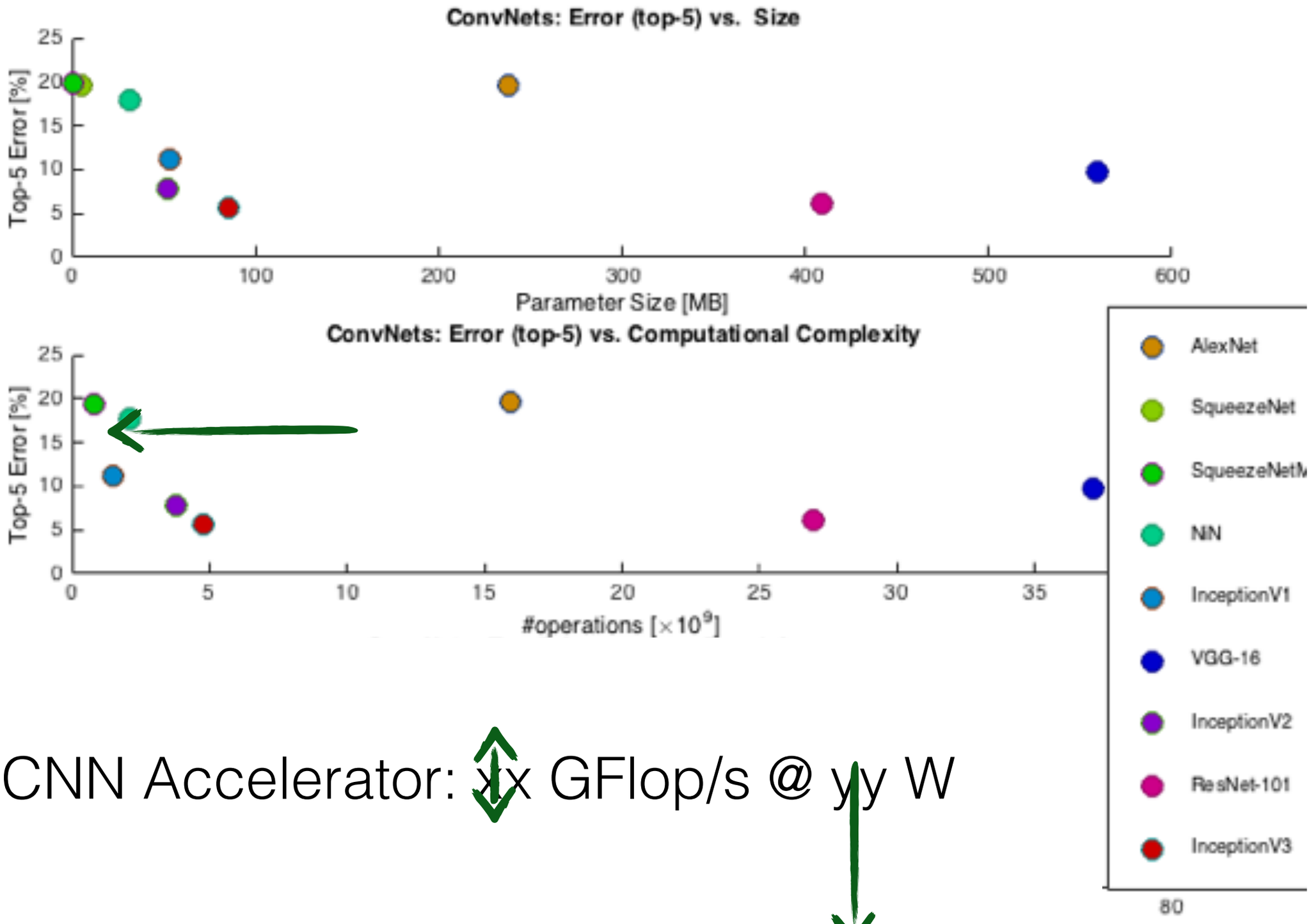
Accuracy vs. Complexity of CNNs



Titan X GPU: 7000 GFlop/s @ 250W

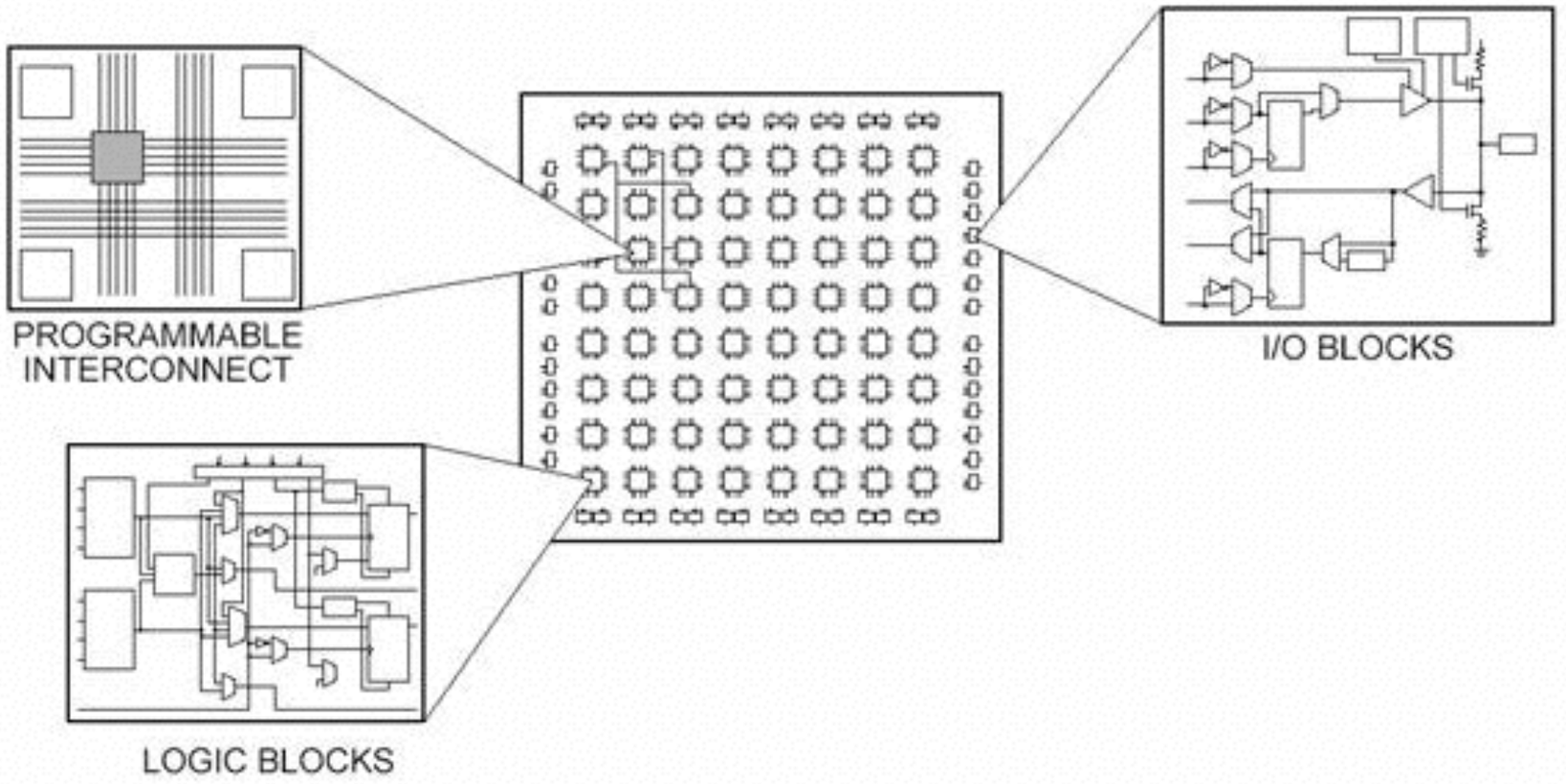
https://www.nvidia.com/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf

What about embedded systems?



CNN Accelerator: xx GFlop/s @ yy W

FPGA = Field Programmable Gate Array

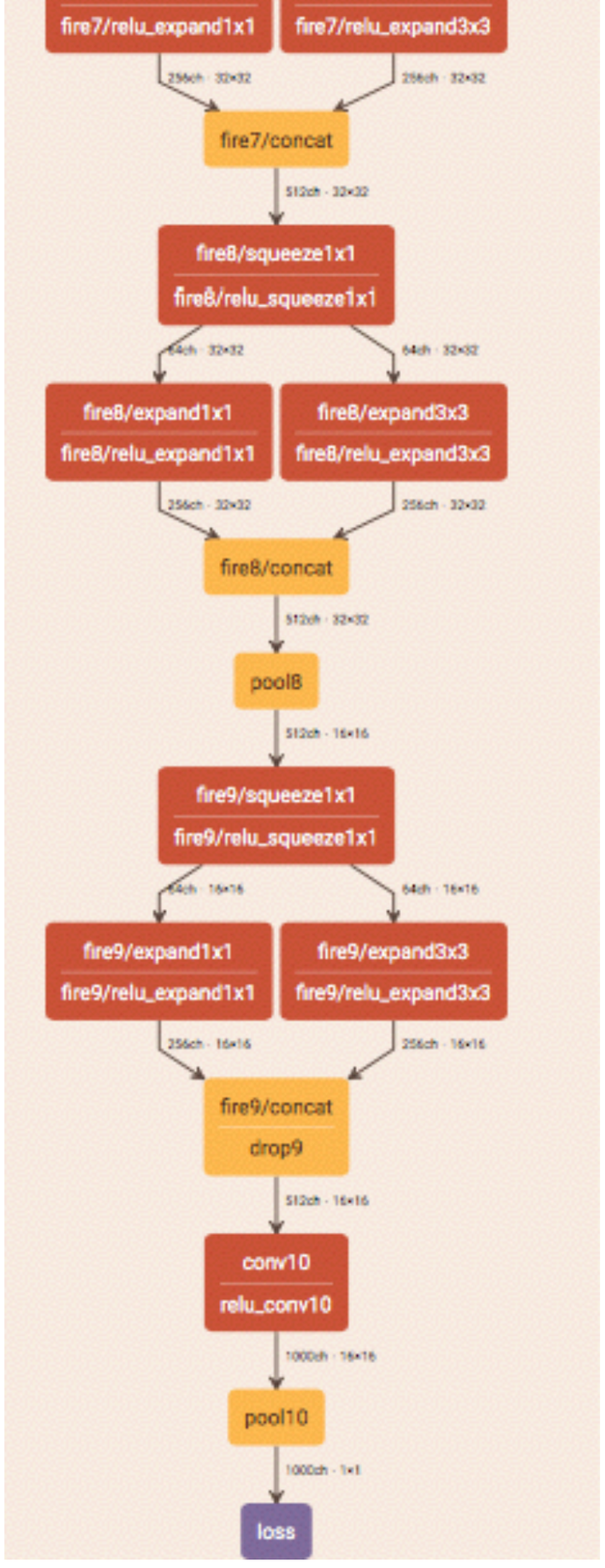


	performance	NREs	Unit cost	TTM
↑	ASIC	ASIC	FPGA	ASIC
	FPGA	FPGA	MICRO	FPGA
	MICRO	MICRO	ASIC	MICRO

ASIC = custom IC, MICRO = microprocessor

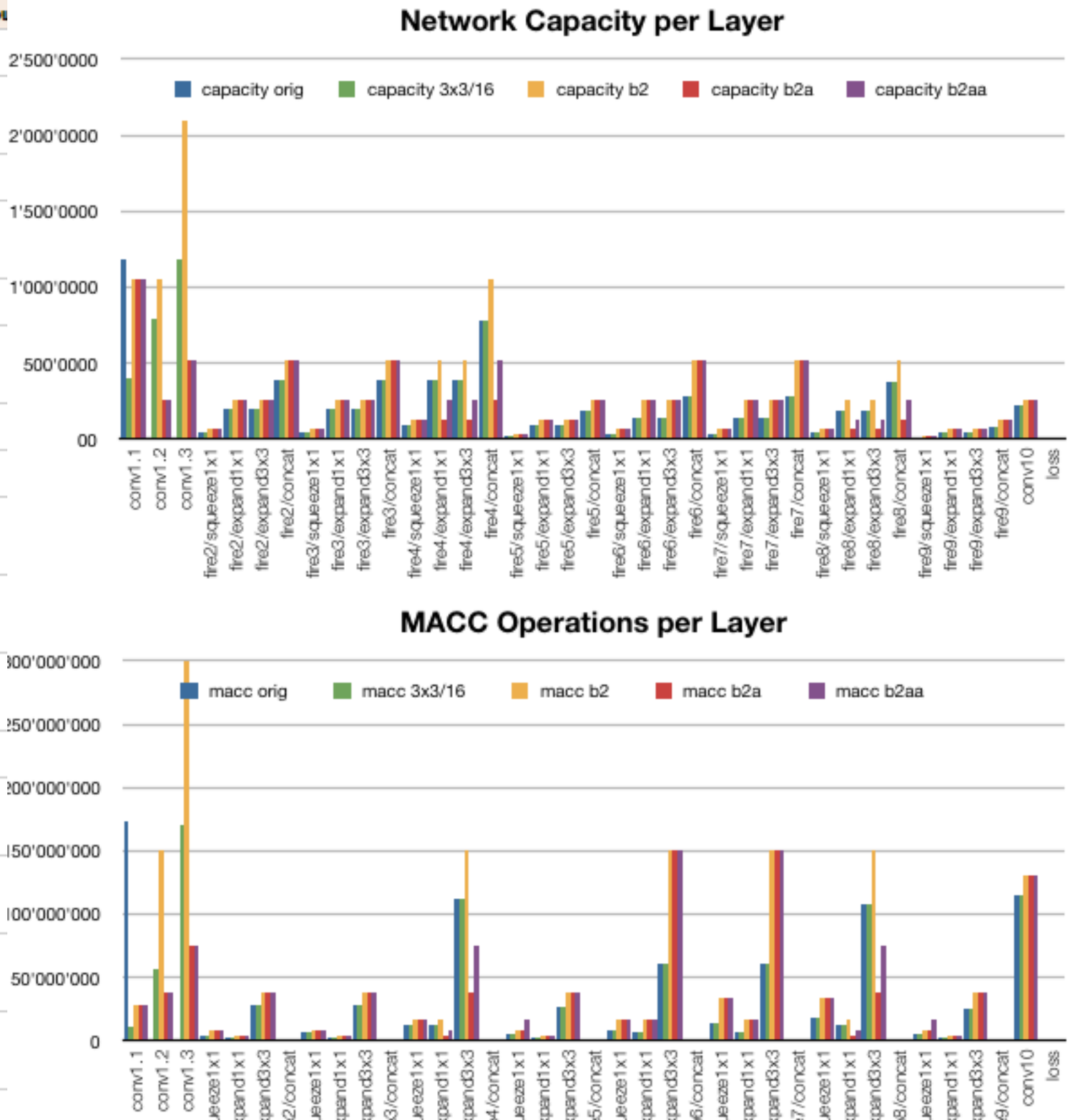
http://www.tutorial-reports.com/computer-science/fpga/overview.php
http://www.ni.com/white-paper/6983/de/

CNN Topologie



CNN Topologie

ID	name	type	ch_in	dim_in	ch_ou
1	data	Data	3	256x256	3
2	conv1.1	Convolution	3	256x256	16
3	relu_conv1.1	ReLU	16	256x256	16
4	conv1.2	Convolution	16	256x256	16
5	relu_conv1.2	ReLU	16	256x256	16
6	conv1.3	Convolution	16	256x256	128
7	relu_conv1.3	ReLU	128	128x128	128
8	pool1	Pooling	128	128x128	128
9	fire2	submodule(6)	128	64x64	128
16	fire3	submodule(6)	128	64x64	128
23	fire4	submodule(6)	128	64x64	256
30	pool4	Pooling	256	64x64	256
31	fire5	submodule(6)	256	32x32	256
38	fire6	submodule(6)	256	32x32	512
45	fire7	submodule(6)	512	32x32	512
52	fire8	submodule(6)	512	32x32	512
59	pool8	Pooling	512	32x32	512



rhea.scs-ad.scs.ch/models/images/classification/new?clone=20160425-150250

fram|alpha dilbert xkcd PwnYouTube PowerYoga After Work Revitalizer MA 14-Day Yoga Challenge Mindfulr

DIGITS New Model Version 3.0.0

New Image Classification Model

Select Dataset

CIFAR10_8
CIFAR10_16
ImageNet_1k_300
ImageNet_1k_256
ImageNet_1k_128

Use client side file

Python Layer File (server side)

Solver Options

Training epochs
30

Snapshot interval (in epochs)
1.0

Validation interval (in epochs)
1.0

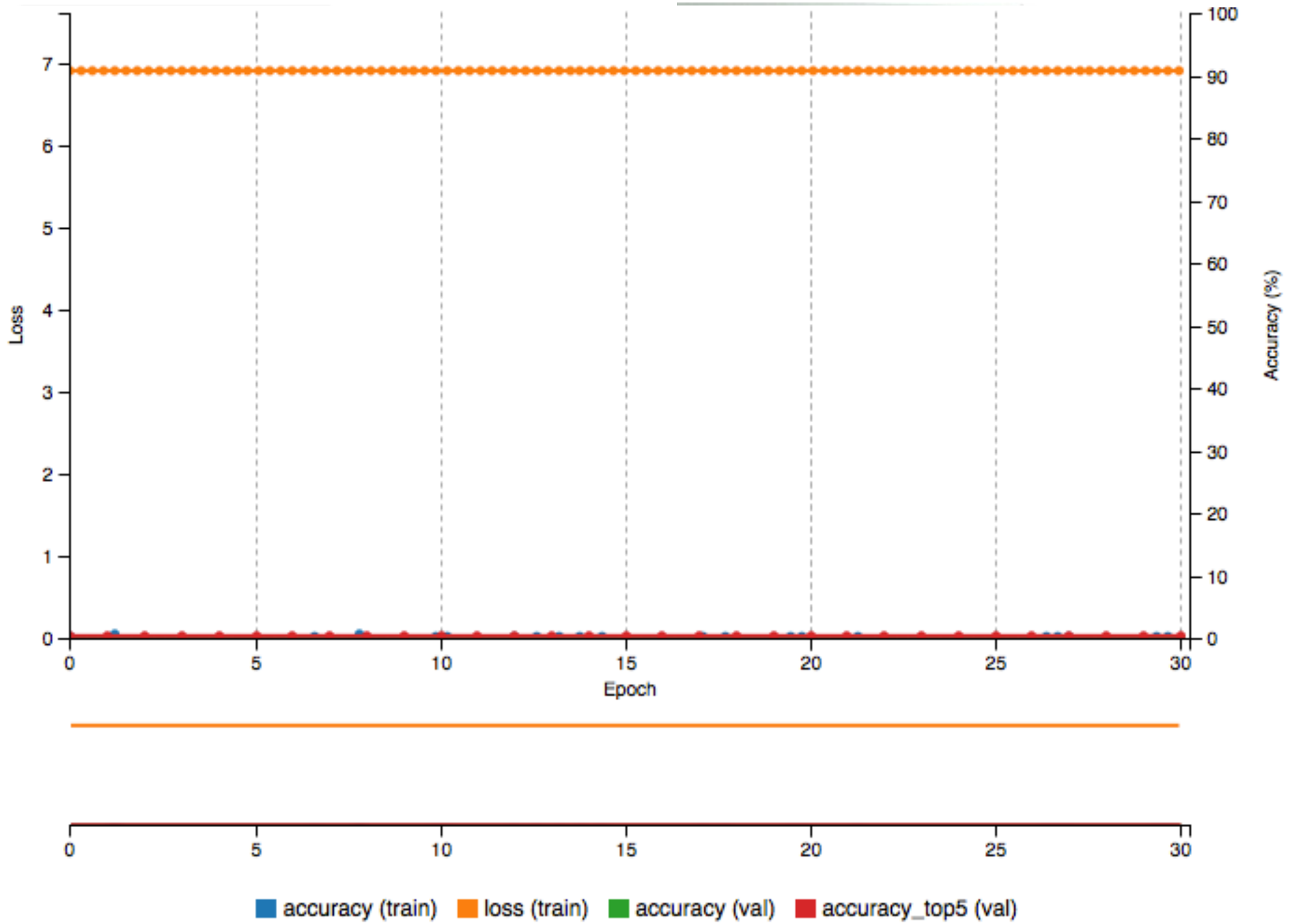
Random seed
42

Batch size
96

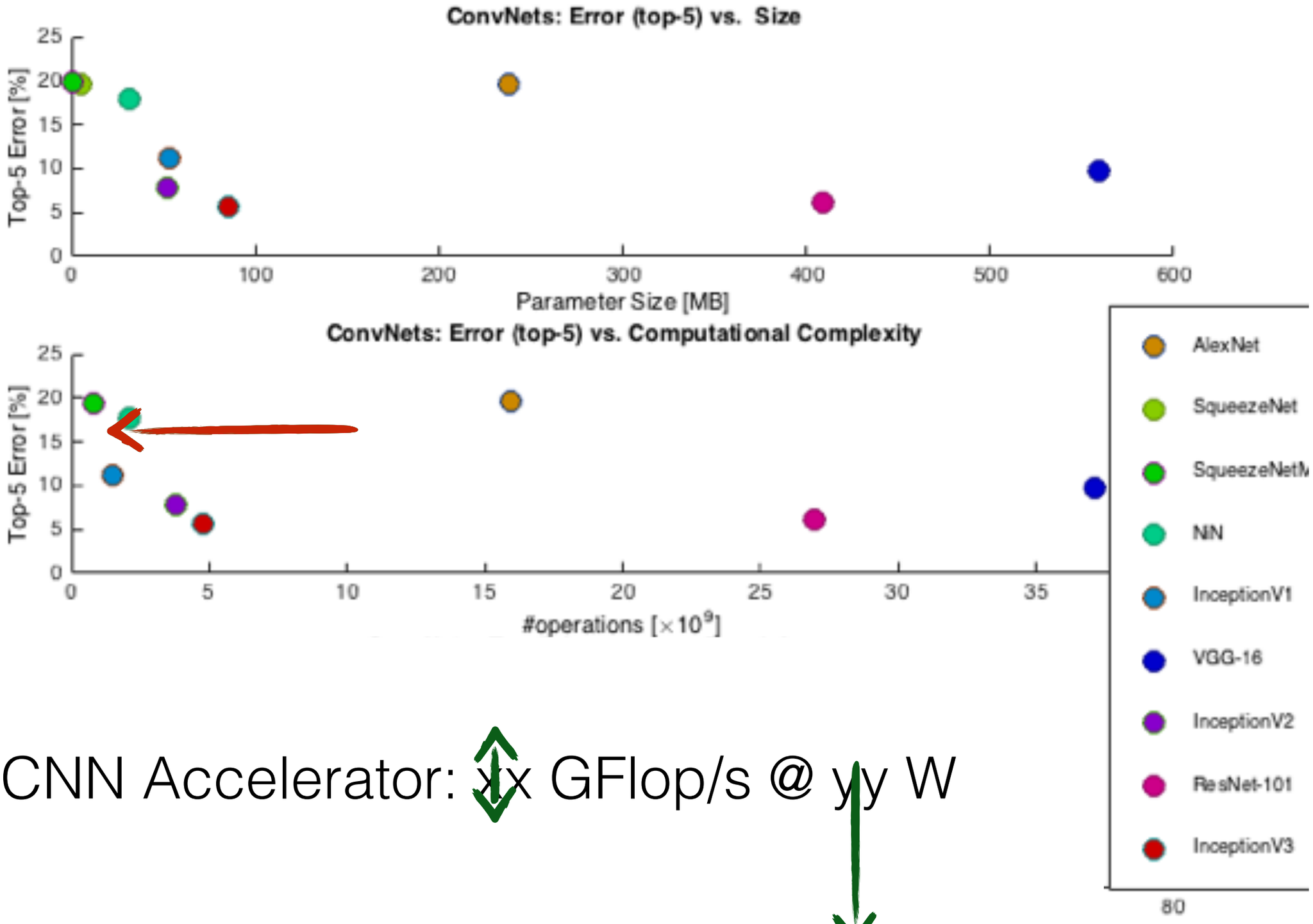
Solver type
Stochastic gradient descent (SGD)

Base Learning Rate
0.01

Show advanced learning rate options

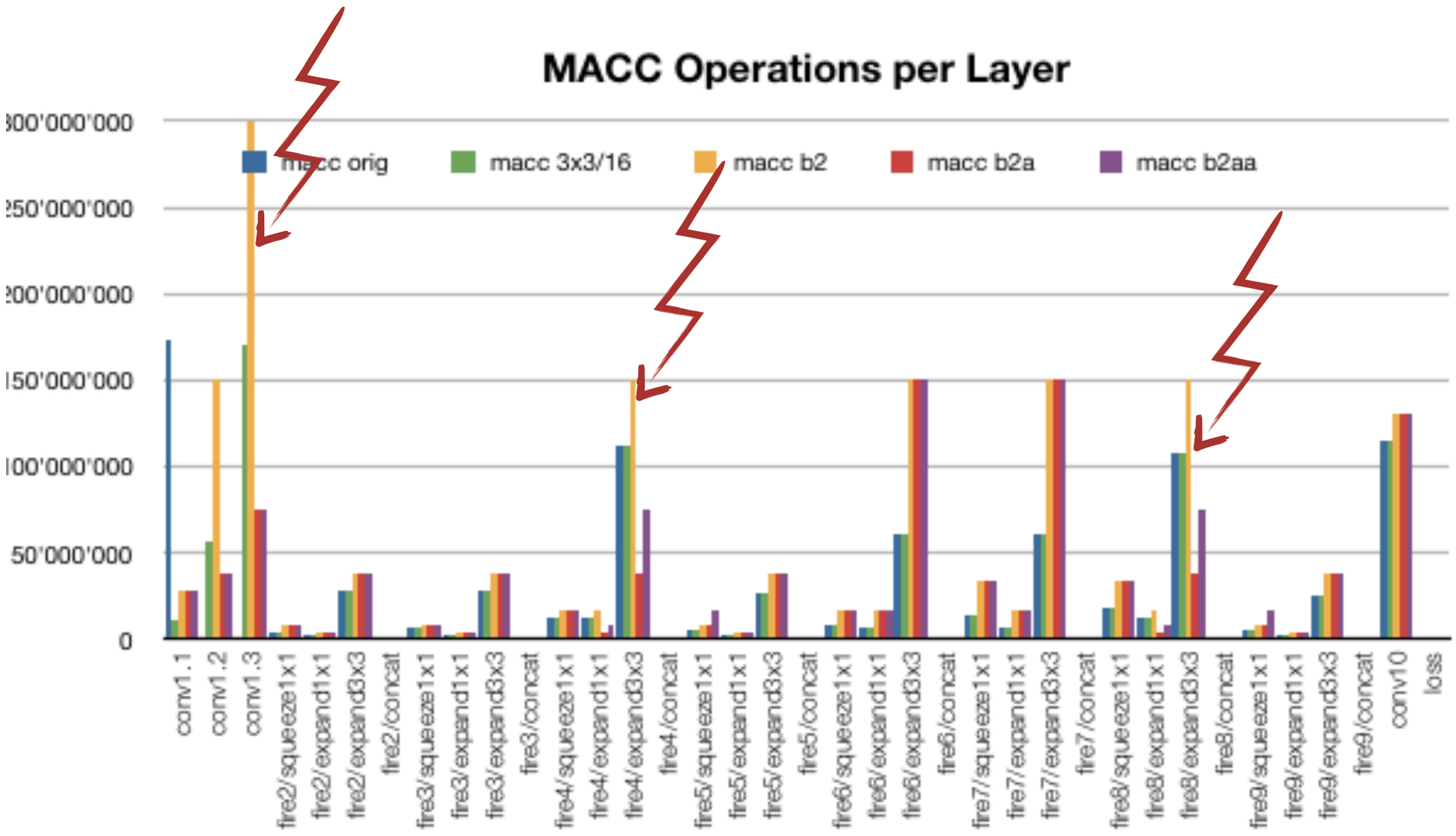


Co-Adaption FPGA / CNN

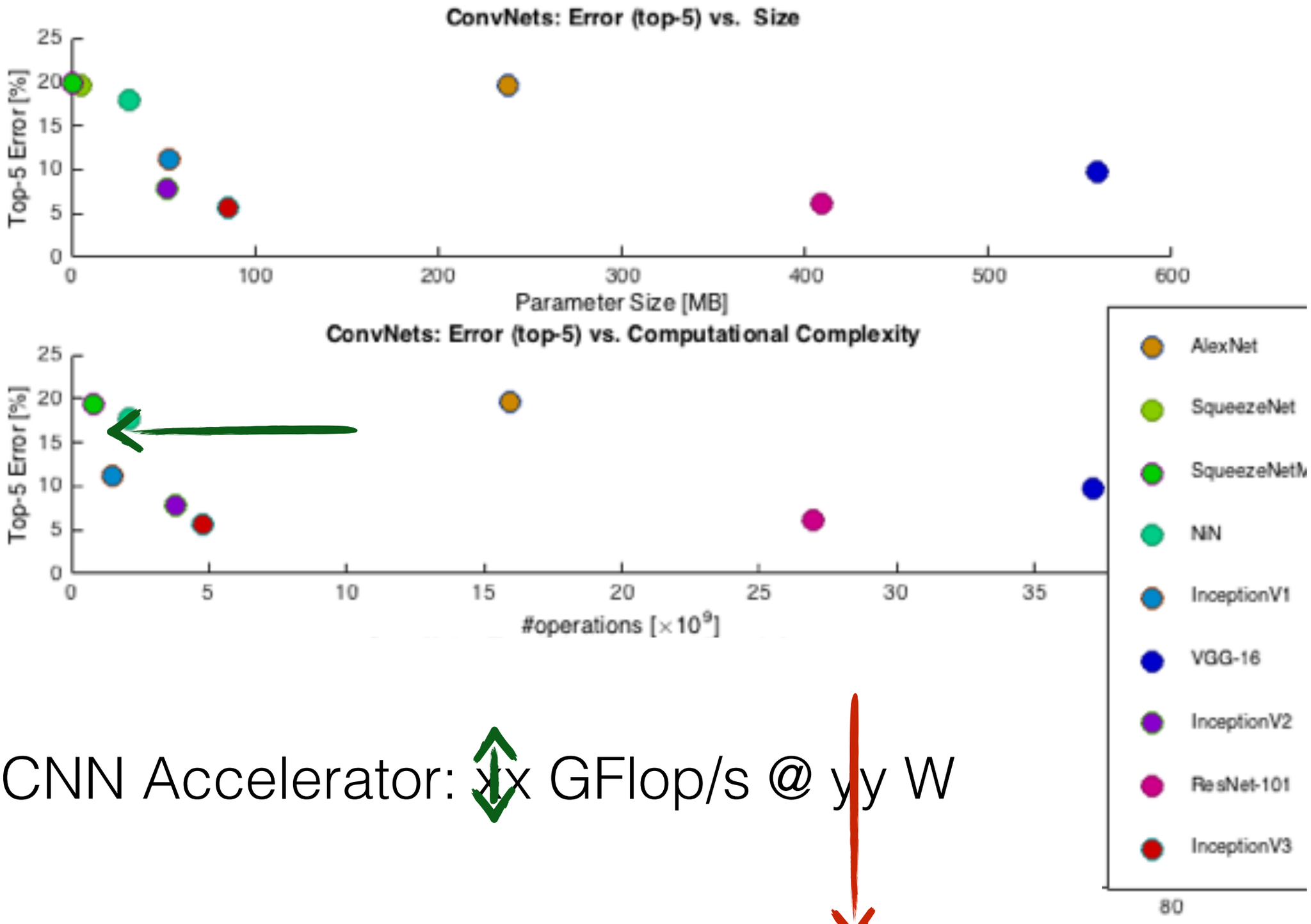


CNN Accelerator: xx GFlop/s @ yy W

Co-Adaption FPGA / CNN

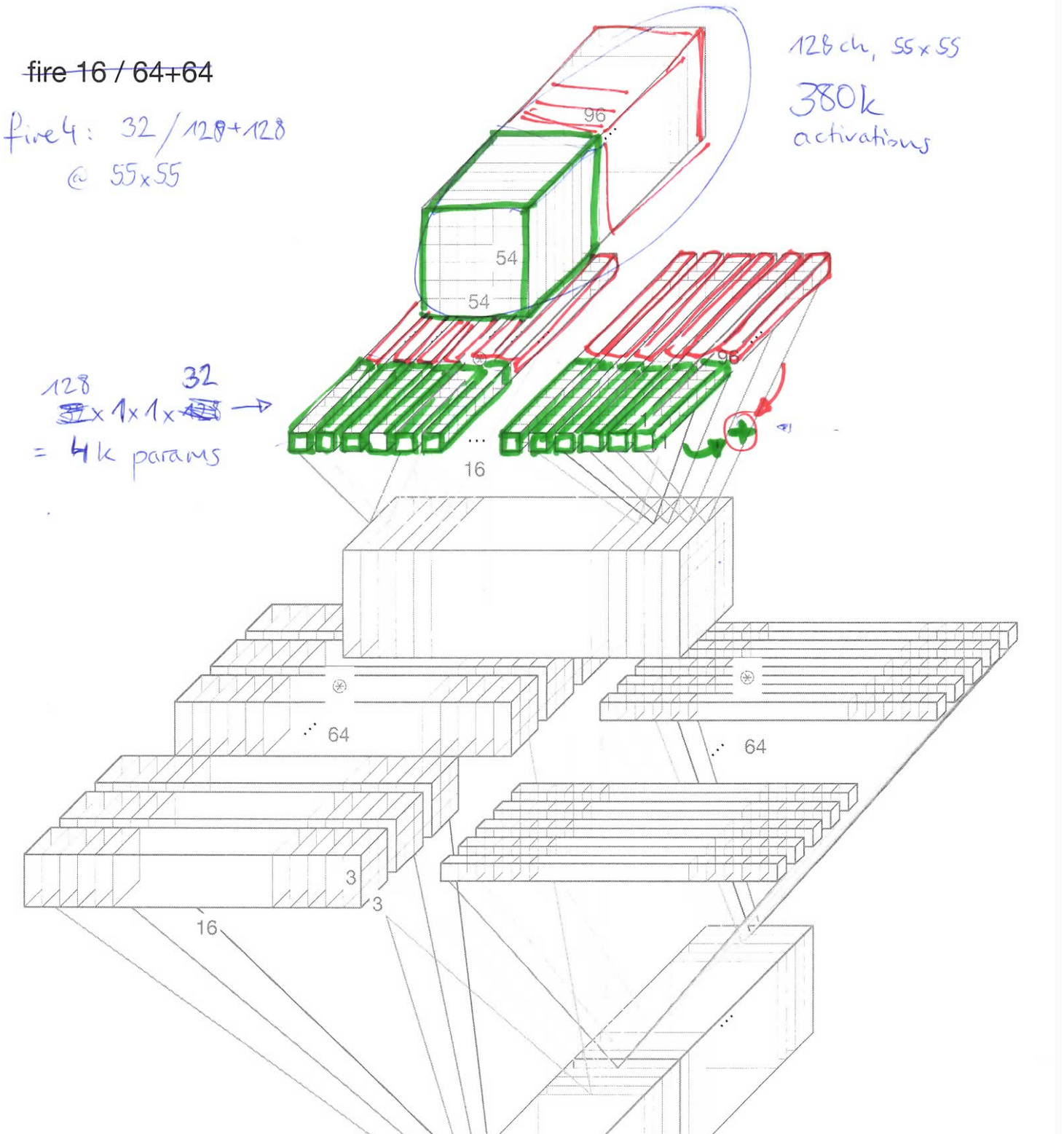


Co-Adaption FPGA / CNN

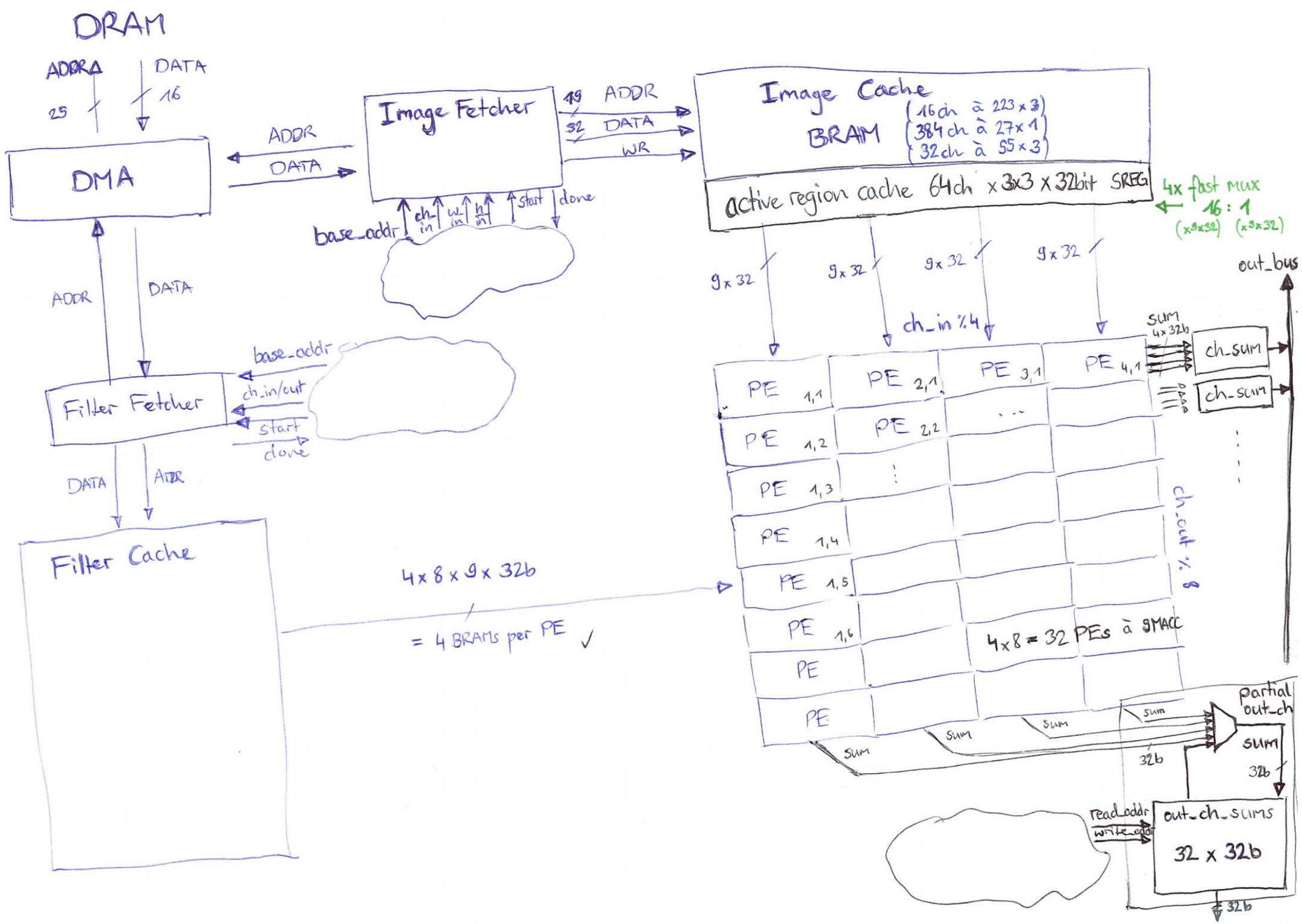


CNN Accelerator: xx GFlop/s @ yy W

Co-Adaption FPGA / CNN



Co-Adaption FPGA / CNN



FPGA Implementation with HLS

HLS = High-Level Synthesis

```
float conv2D = 0;
kernel_t i, j;

for (j = 0; j < 3; j++) {
#pragma HLS unroll
  for (i = 0; i < 3; i++) {
#pragma HLS unroll
    int filt_addr = ci*(ch_out*9)+co*9+j*3+i;
    conv2D += FILTER_CACHE[filter_addr]*
              ACTIVE_AREA[current_aa][j*3+i];
  }
}
```

C/C++ Code

Vivado HLS

```
    NUM_STAGE => 4,
    din0_WIDTH => 32,
    din1_WIDTH => 32,
    dout_WIDTH => 32)
port map (
    clk => ap_clk,
    reset => ap_rst,
    din0 => img_in_q1,
    din1 => filt_in_q1,
    ce => grp_fu_314_ce,
    dout => grp_fu_314_p2);



-- the current state (ap_CS_fsm) of the state machine. --
ap_CS_fsm_assign_proc : process(ap_clk)
begin
    if (ap_clk'event and ap_clk = '1') then
        if (ap_rst = '1') then
            ap_CS_fsm <= ap_ST_pp0_stg0_fsm_0;
        else
            ap_CS_fsm <= ap_NS_fsm;
        end if;
    end if;
end process;


-- ap_reg_ppiten_pp0_it0_preg assign process. --
ap_reg_ppiten_pp0_it0_preg_assign_proc : process(ap_clk)
begin
    if (ap_clk'event and ap_clk =  '1') then
        if (ap_rst = '1') then
            ap_reg_ppiten_pp0_it0_preg <= ap_const_logic_0;
        else
            if (((ap_const_logic_1 = ap_sig_cseq_ST_pp0_stg0_fsm_0) and not(((ap_const_log
                ap_reg_ppiten_pp0_it0_preg <= ap_start;
            end if;
        end if;
    end if;
end process;


-- ap_reg_ppiten_pp0_it1 assign process. --
ap_reg_ppiten_pp0_it1_assign_proc : process(ap_clk)
begin
    if (ap_clk'event and ap_clk =  '1') then
        if (ap_rst = '1') then
            ap_reg_ppiten_pp0_it1 <= ap_const_logic_0;
        else
            if ((ap_const_logic_1 = ap_sig_cseq_ST_pp0_stg4_fsm_4)) then
                ap_reg_ppiten_pp0_it1 <= ap_reg_ppiten_pp0_it0;
            end if;
        end if;
    end if;
end process;


-- ap_reg_ppiten_pp0_it10 assign process. --
ap_reg_ppiten_pp0_it10_assign_proc : process(ap_clk)
begin
    if (ap_clk'event and ap_clk =  '1') then
```

VHDL

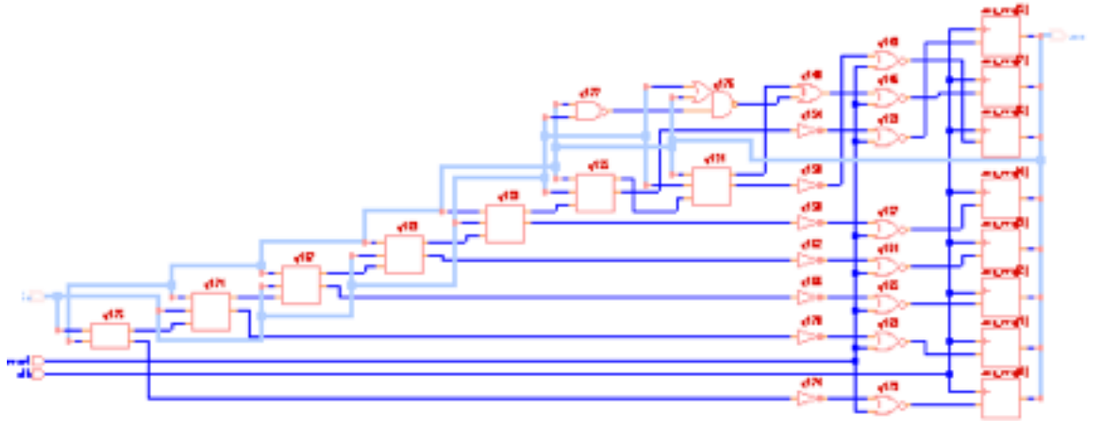
FPGA Implementation with HLS

HLS = High-Level Synthesis

```
do_macc_fmul_32ns_32ns_32_4_max_dsp_U4 : component do_macc_fmul_32ns_32ns_32_4_max_dsp
generic map (
    ID => 1,
    NUM_STAGE => 4,
    din0_WIDTH => 32,
    din1_WIDTH => 32,
    dout_WIDTH => 32)
port map (
    clk => ap_clk,
    reset => ap_rst,
    din0 => img_in_q1,
    din1 => filt_in_q1,
    ce => grp_fu_314_ce,
    dout => grp_fu_314_p2);
```

```
-- the current state (ap_CS_fsm) of the state machine. --
ap_CS_fsm_assign_proc : process(ap_clk)
begin
    if (ap_clk'event and ap_clk =  '1') then
        if (ap_rst = '1') then
            ap_CS_fsm <= ap_ST_pp0_stg0_fsm_0;
        else
            ap_CS_fsm <= ap_NS_fsm;
        end if;
    end if;
end process;


-- ap_reg_ppiten_pp0_it0_preg assign process. --
ap_reg_ppiten_pp0_it0_preg_assign_proc : process(ap_clk)
begin
    if (ap_clk'event and ap_clk =  '1') then
        if (ap_rst = '1') then
            ap_reg_ppiten_pp0_it0_preg <= ap_const_logic_0;
        else
            if (((ap_const_logic_1 = ap_sig_cseq_ST_pp0_stg0_fsm_0) and not(((ap_const_logic_1 = ap_reg_ppiten_pp0_it0) and (ap_start = ap_const_logic_0))))) then
                ap_reg_ppiten_pp0_it0_preg <= ap_start;
            end if;
        end if;
    end if;
end process;


-- ap_reg_ppiten_pp0_it1 assign process. --
ap_reg_ppiten_pp0_it1_assign_proc : process(ap_clk)
begin
    if (ap_clk'event and ap_clk =  '1') then
        if (ap_rst = '1') then
            ap_reg_ppiten_pp0_it1 <= ap_const_logic_0;
        else
            if ((ap_const_logic_1 = ap_sig_cseq_ST_pp0_stg4_fsm_4)) then
                ap_reg_ppiten_pp0_it1 <= ap_reg_ppiten_pp0_it0;
            end if;
        end if;
    end if;
end process;


-- ap_reg_ppiten_pp0_it10 assign process. --
ap_reg_ppiten_pp0_it10_assign_proc : process(ap_clk)
begin
    if (ap_clk'event and ap_clk =  '1') then
        if (ap_rst = '1') then
            ap_reg_ppiten_pp0_it10 <= ap_const_logic_0;
        else
            if (((ap_const_logic_1 = ap_sig_cseq_ST_pp0_stg3_fsm_3) and not((ap_const_logic_1 = ap_reg_ppiten_pp0_it9)))) then
                ap_reg_ppiten_pp0_it10 <= ap_const_logic_0;
            elsif ((ap_const_logic_1 = ap_sig_cseq_ST_pp0_stg4_fsm_4)) then
                ap_reg_ppiten_pp0_it10 <= ap_reg_ppiten_pp0_it9;
            end if;
        end if;
    end if;
end process;


-- ap_reg_ppiten_pp0_it2 assign process. --
ap_reg_ppiten_pp0_it2_assign_proc : process(ap_clk)
begin
    if (ap_clk'event and ap_clk =  '1') then
        if (ap_rst = '1') then
            ap_reg_ppiten_pp0_it2 <= ap_const_logic_0;
        else
            if ((ap_const_logic_1 = ap_sig_cseq_ST_pp0_stg4_fsm_4)) then
                ap_reg_ppiten_pp0_it2 <= ap_reg_ppiten_pp0_it1;
            end if;
        end if;
    end if;
end process;


-- ap_reg_ppiten_pp0_it3 assign process. --
ap_reg_ppiten_pp0_it3_assign_proc : process(ap_clk)
begin
    if (ap_clk'event and ap_clk =  '1') then
        if (ap_rst = '1') then
            ap_reg_ppiten_pp0_it3 <= ap_const_logic_0;
        else
            if ((ap_const_logic_1 = ap_sig_cseq_ST_pp0_stg4_fsm_4)) then
                ap_reg_ppiten_pp0_it3 <= ap_reg_ppiten_pp0_it2;
            end if;
        end if;
    end if;
end process;
```

RTL Synthesis →



FPGA Configuration (“bit-stream”) ↓

config.bit

FPGA Implementation with HLS

… in progress …

Demo on GPU:
https://rhea.scs-ad.scs.ch/webcam-demo