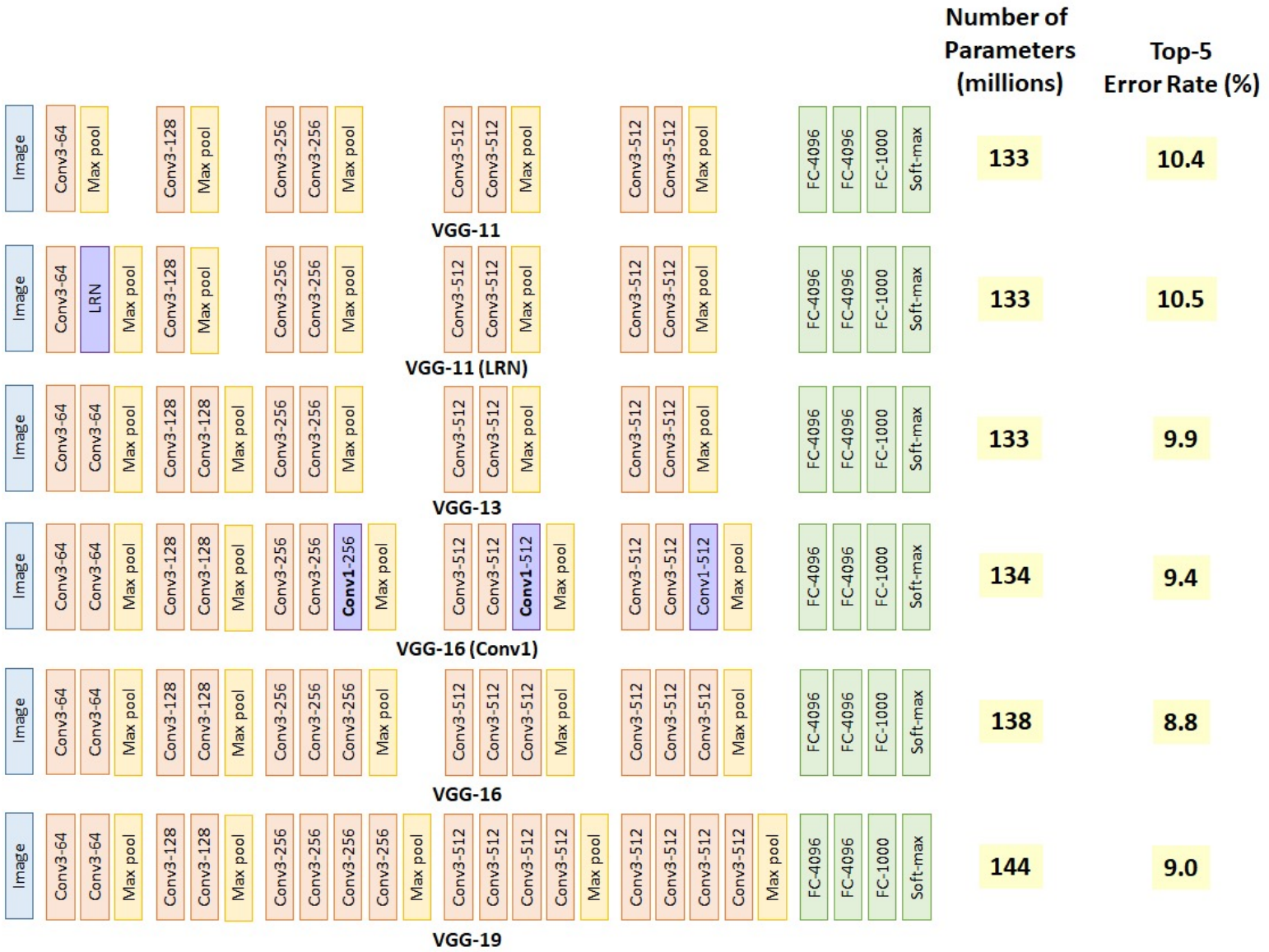
# ECE284 Fall 21 W3S1

## Low-power VLSI Implementation for Machine Learning

Prof. Mingu Kang

# VGGNet



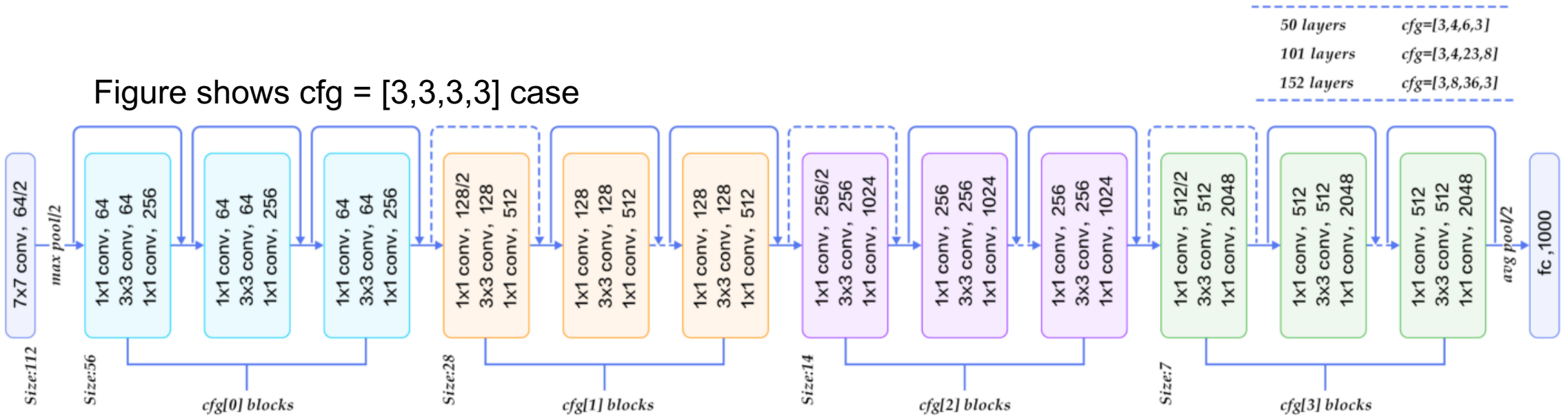

- 1st runner-up in ImageNet challenge in 2014
- Very deep (19-layers) network
- Small filters (3 X 3) used

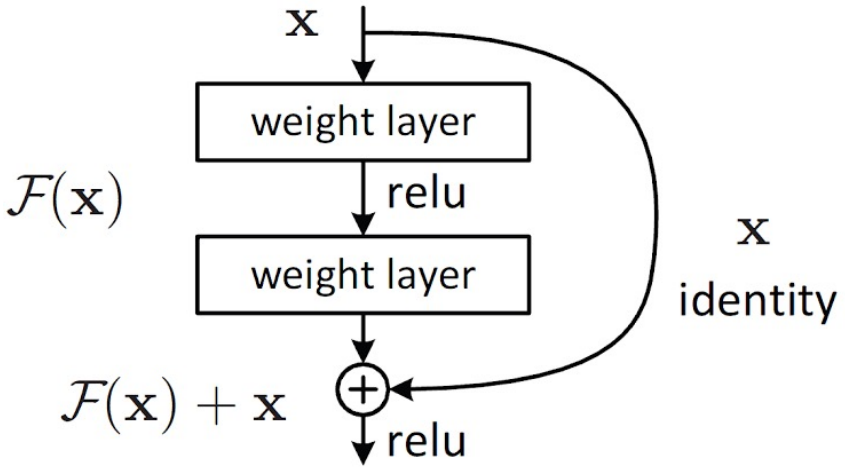
"Very Deep Convolutional Networks for Large-Scale Image Recognition"
https://arxiv.org/abs/1409.1556

Figure:
https://towardsdatascience.com/cnn-architectures-a-deep-dive-a99441d18049

# ResNet: Residual Network

Figure shows cfg = [3,3,3,3] case

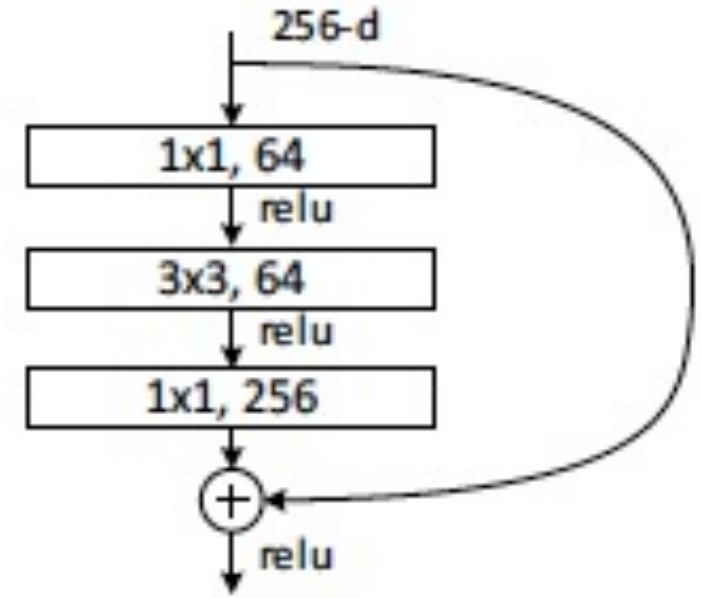



- 1st runner-up in ImageNet challenge in 2015 (3.57% (top-5 error rate))
- Residual network employed to enable very deep neural network without degradation in performance

"Deep Residual Learning for Image Recognition" https://arxiv.org/abs/1512.03385
Figure: https://towardsdatascience.com/cnn-architectures-a-deep-dive-a99441d18049

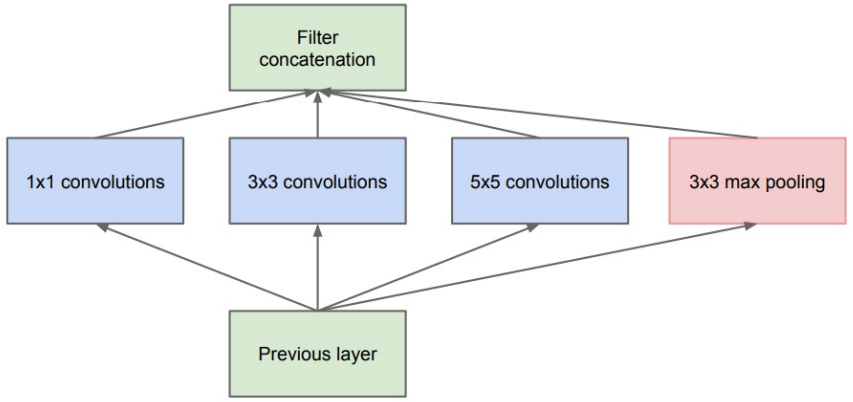
# ResNet: bottleneck layer




- 1x1 convolutions (bottleneck layer) are used to compute reductions before the expensive 3x3 and 5x5 convolutions.

# GoogleNet (Inception, 2015)

Different kernel size needed

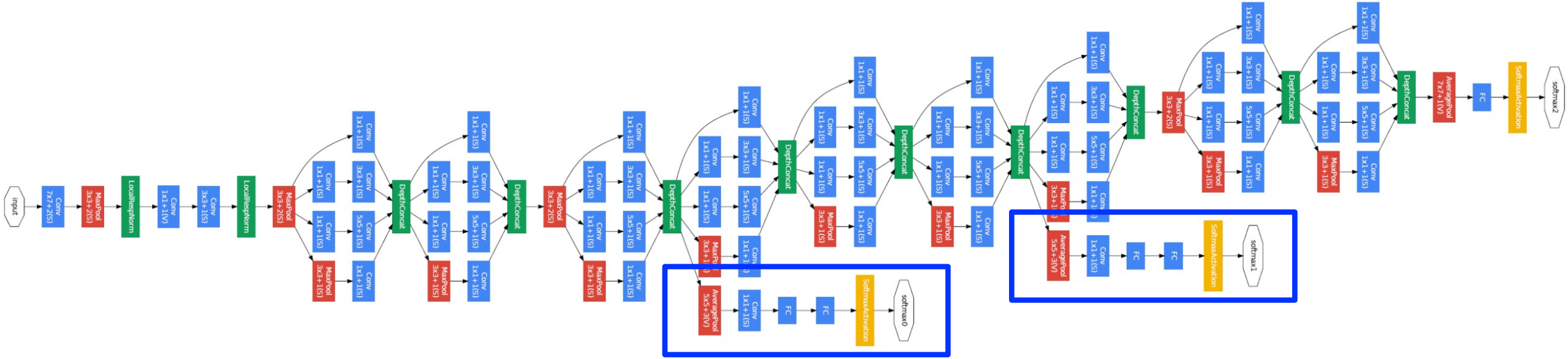



Inception module

- Choice of right kernel size is important.
- A larger kernel is good for global information search whereas a smaller kernel is preferred for local information.
- Inception layer (GoogleNet) introduced

"Going Deeper with Convolutions" https://arxiv.org/abs/1409.4842
https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

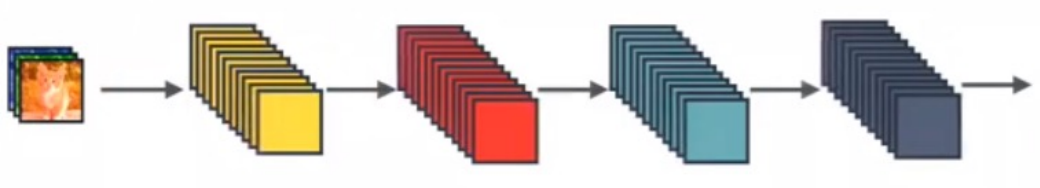
# GoogleNet (Inception, 2015)



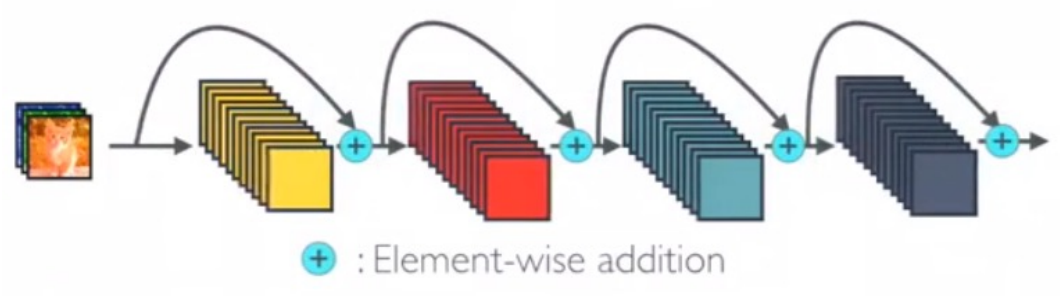

- Intermediate classification layers (used only during the training)
- The layer makes its own decision and its loss are summed for the total loss
- This is to prevent the gradient vanishing problem in the deep network

"Going Deeper with Convolutions" https://arxiv.org/abs/1409.4842

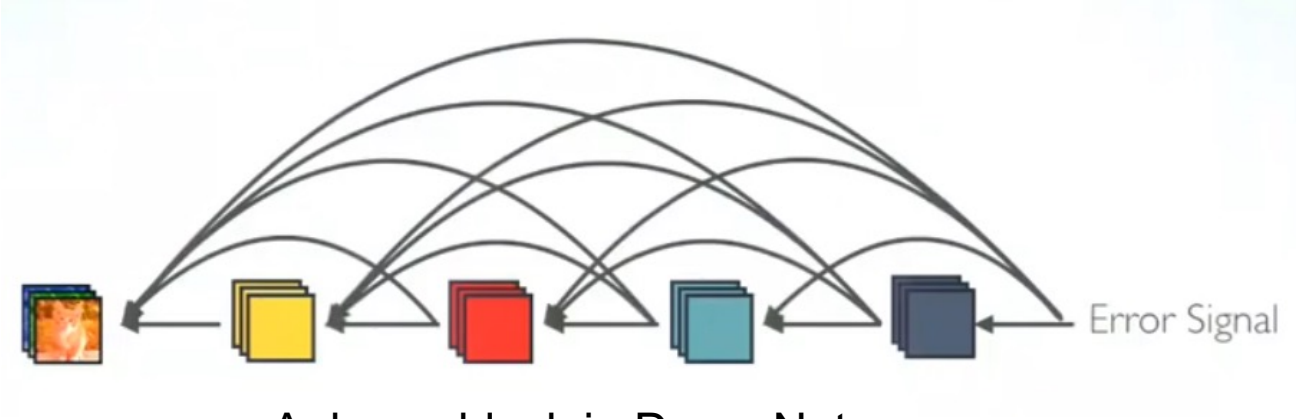
# DenseNet (2017 CVPR)



Standard deep learning



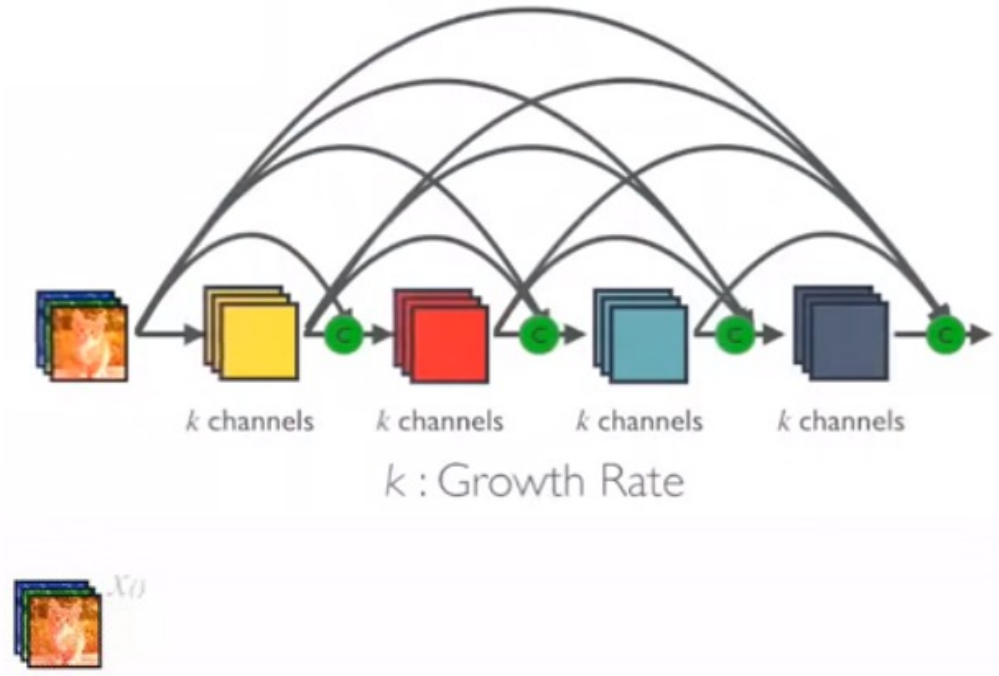

Residual network

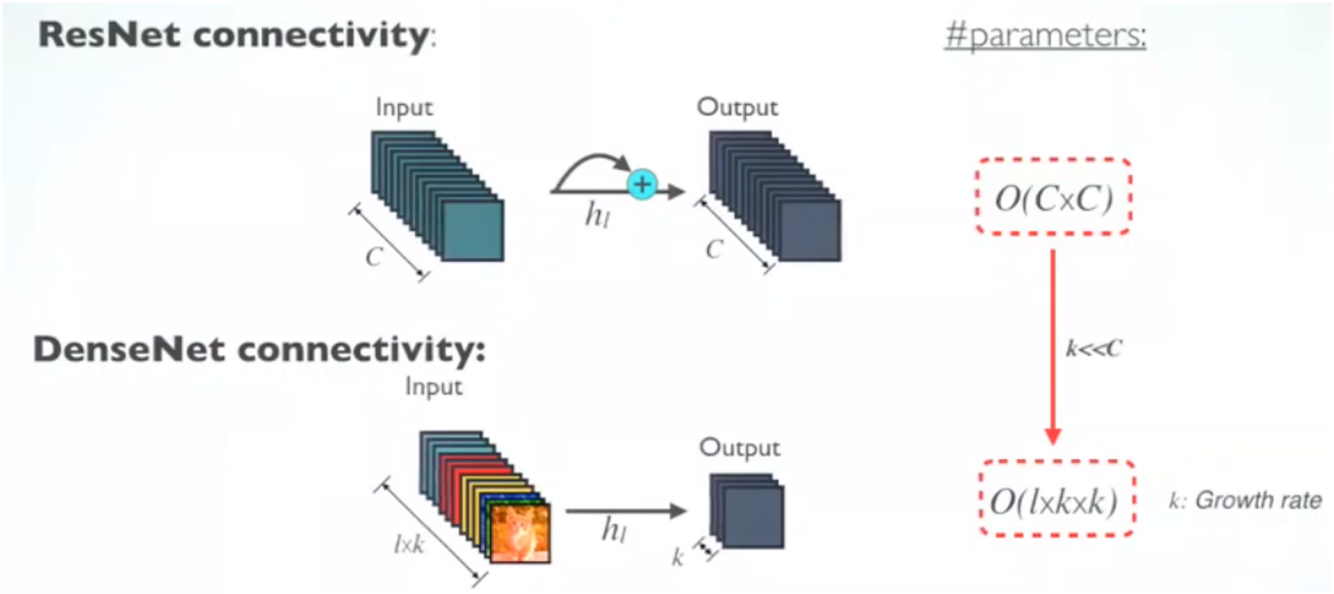



A dense block in DenseNet

"Densely Connected Convolutional Networks" https://arxiv.org/abs/1608.06993
Figure: https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

# DenseNet: Channel-wise concatenation (2017 CVPR)



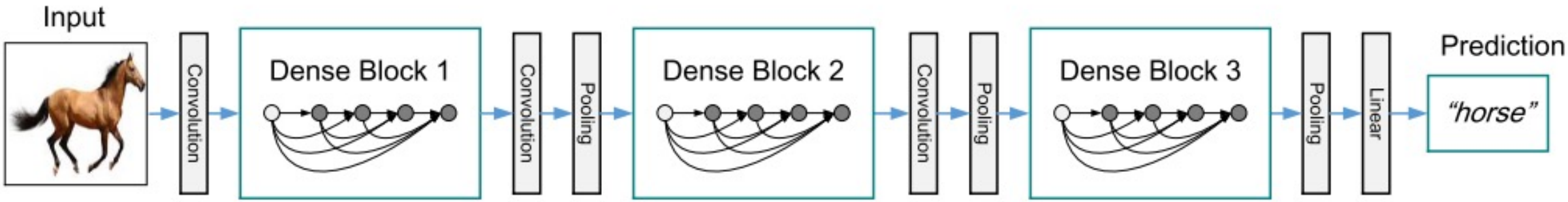




Channel-wise concatenation

- Each layer receives feature maps from all preceding layers
- So, network can be thinner and compact, i.e. number of channels can be fewer.
- Vanishing gradient issue alleviated

"Densely Connected Convolutional Networks" https://arxiv.org/abs/1608.06993
Figure: https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

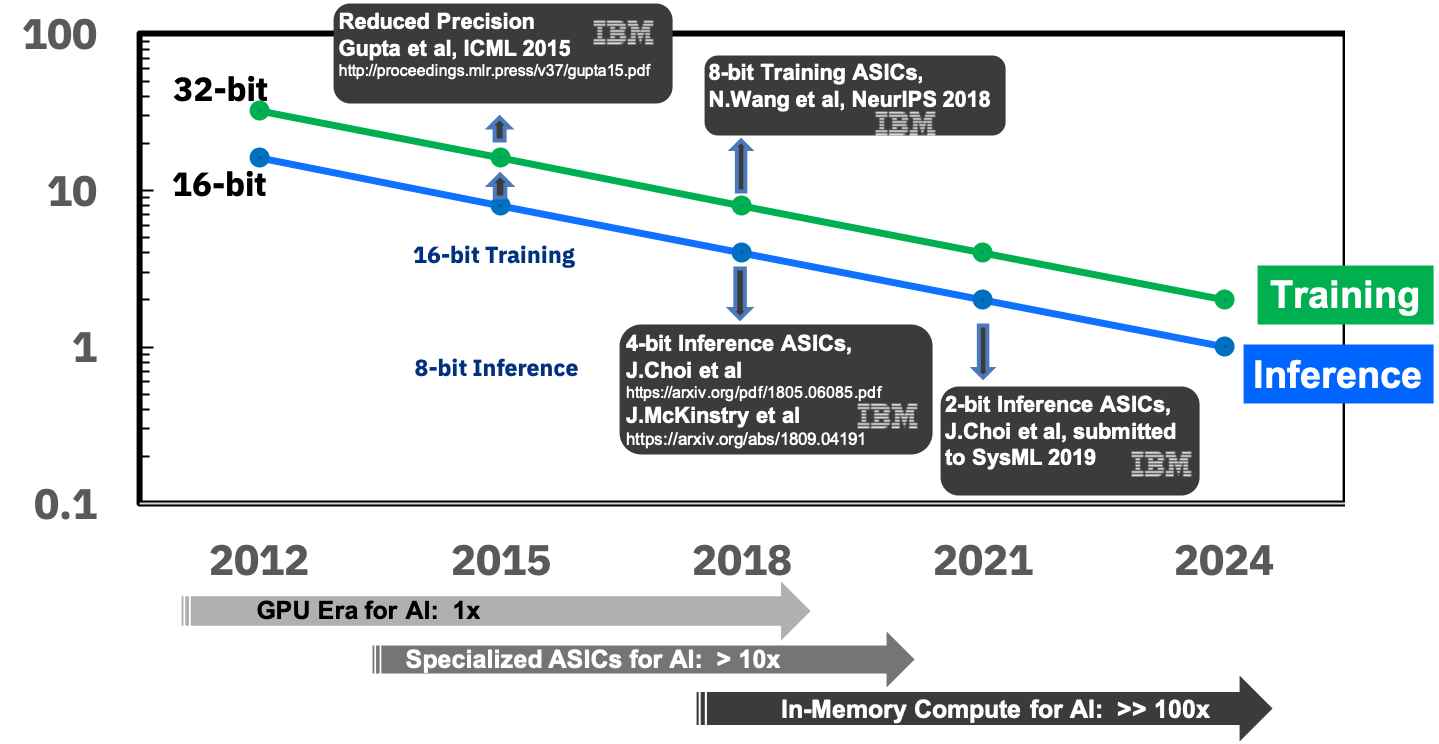
# DenseNet

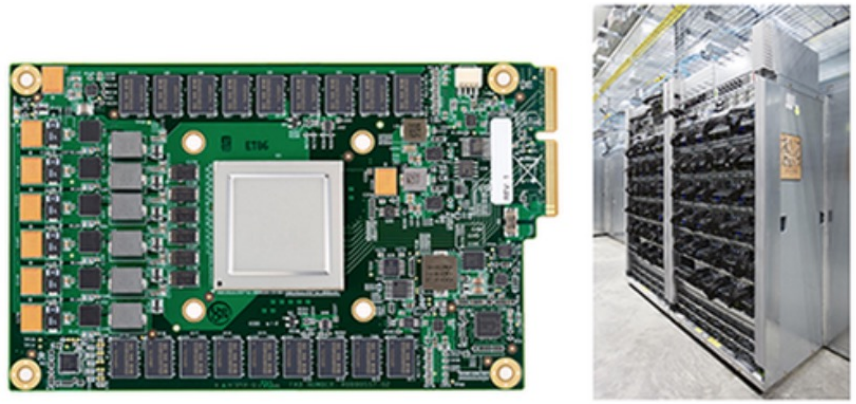



- Same feature map size maintained inside the same dense block (by using padding)
- This allows the concatenation.
- Between dense blocks the size is reduced via convolution and pooling operation

# [CODE] VGGNet ResNet Training (HW3_prob1)

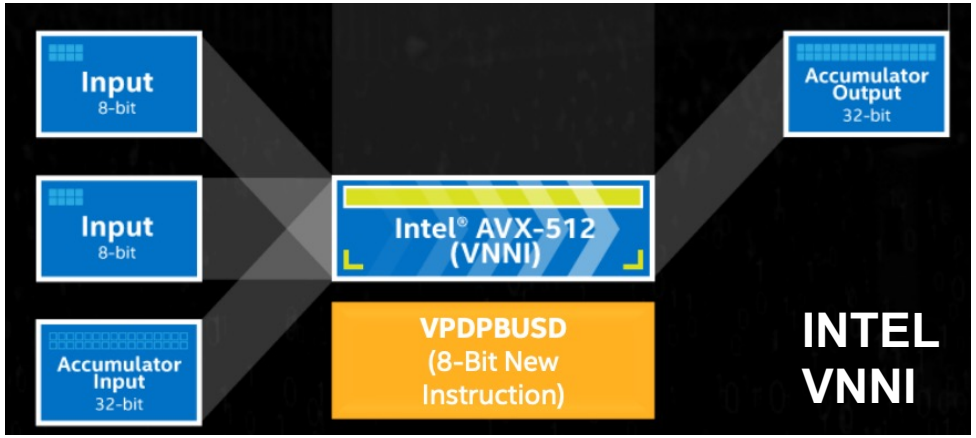
# Number Representation (Quantization) in ML Hardware



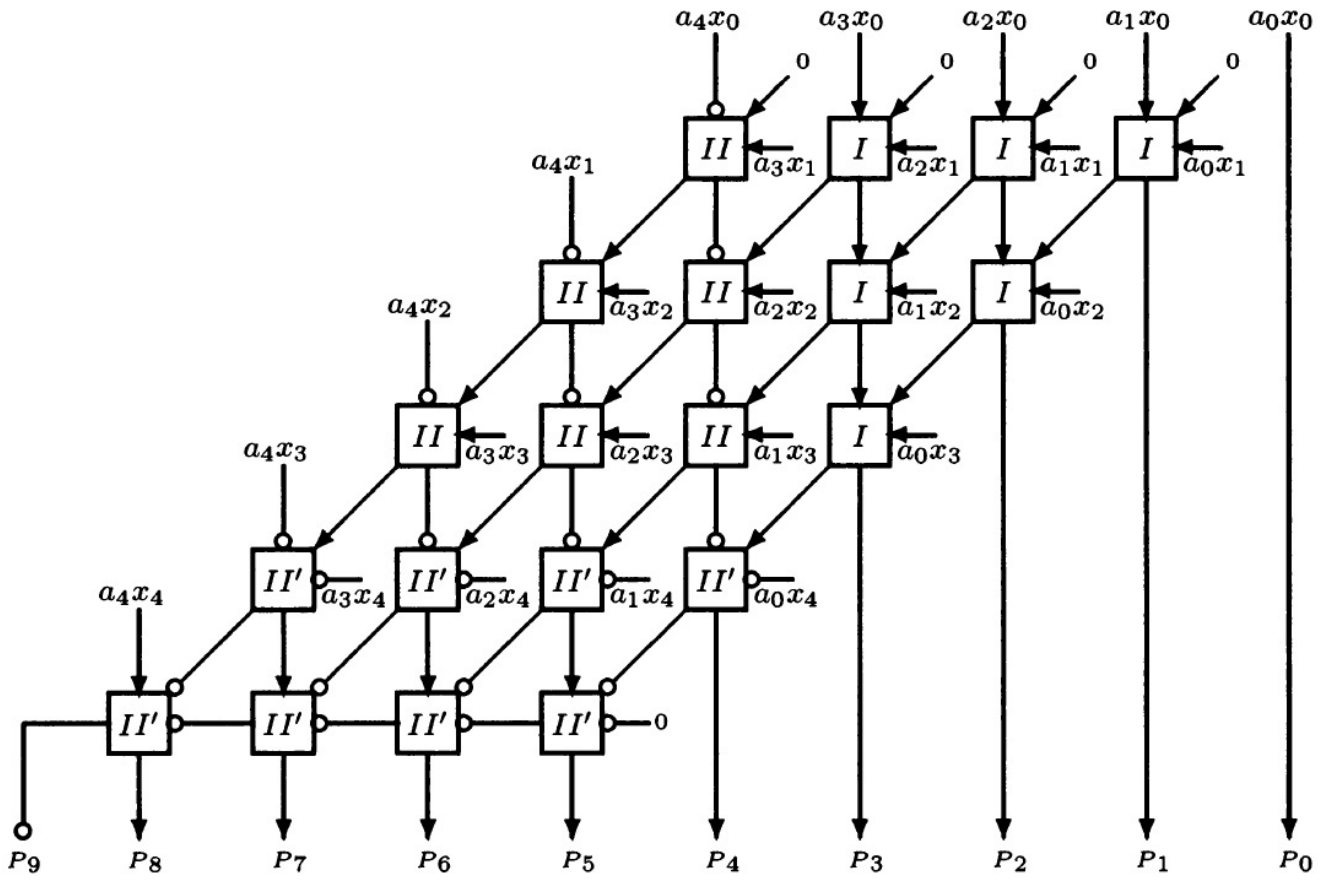
Google's first Tensor Processing Unit (TPU) on a printed circuit board (left); TPUs deployed in a Google datacenter

"A TPU contains 65,536 8-bit integer multipliers."




https://www.ibm.com/blogs/research/2018/12/8-bit-precision-training/
https://www.intel.com/content/dam/www/public/us/en/documents/product-overviews/dl-boost-product-overview.pdf
https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

# Why number representation matters?

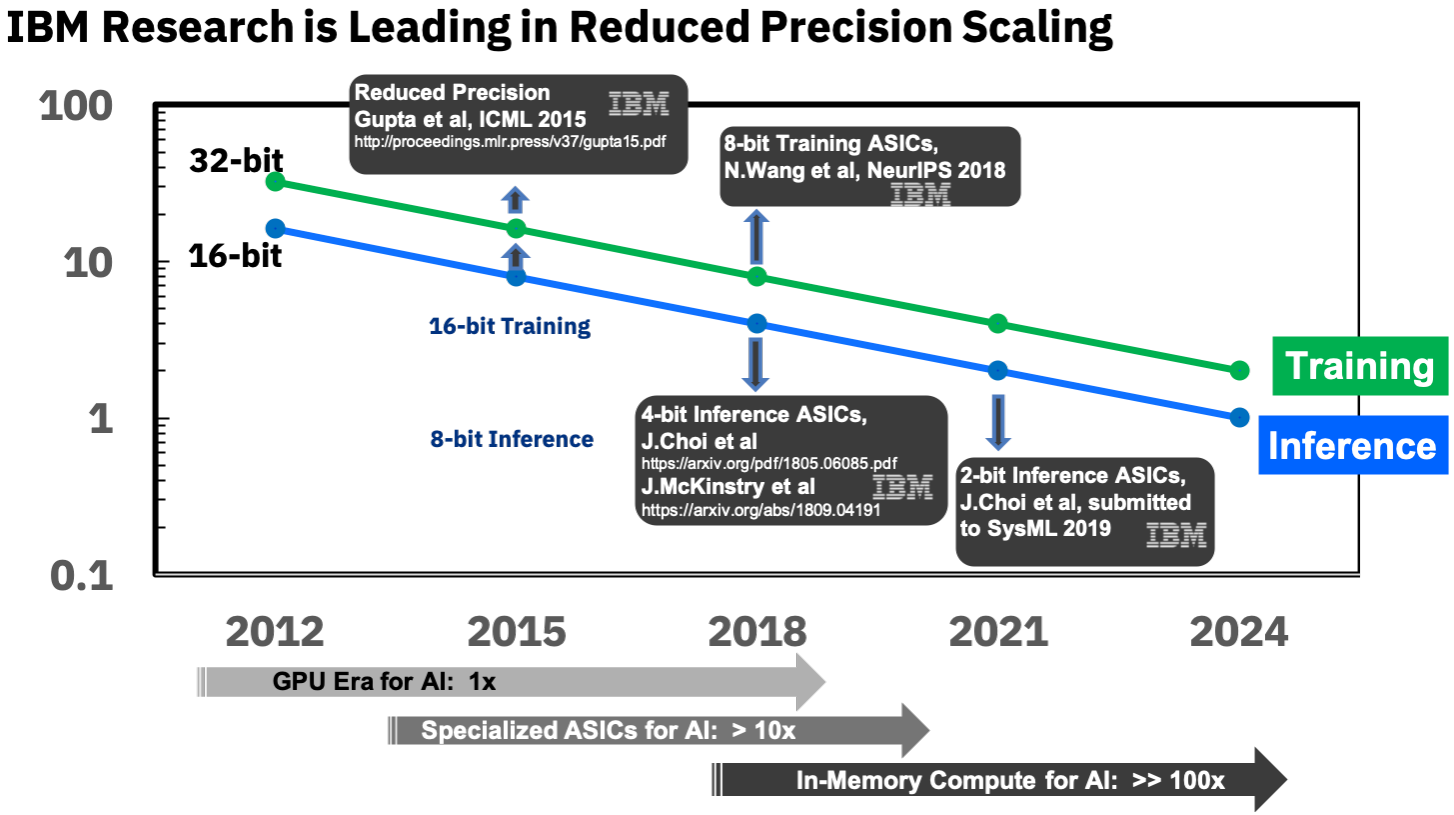



4bit x 4bit array multiplier

- Intuitively, by reducing the input bit precision by a factor of S, the hardware complexity, e.g., area, energy, and power, reduces by $S^2$.

- Simple and effective solution to improve efficiency

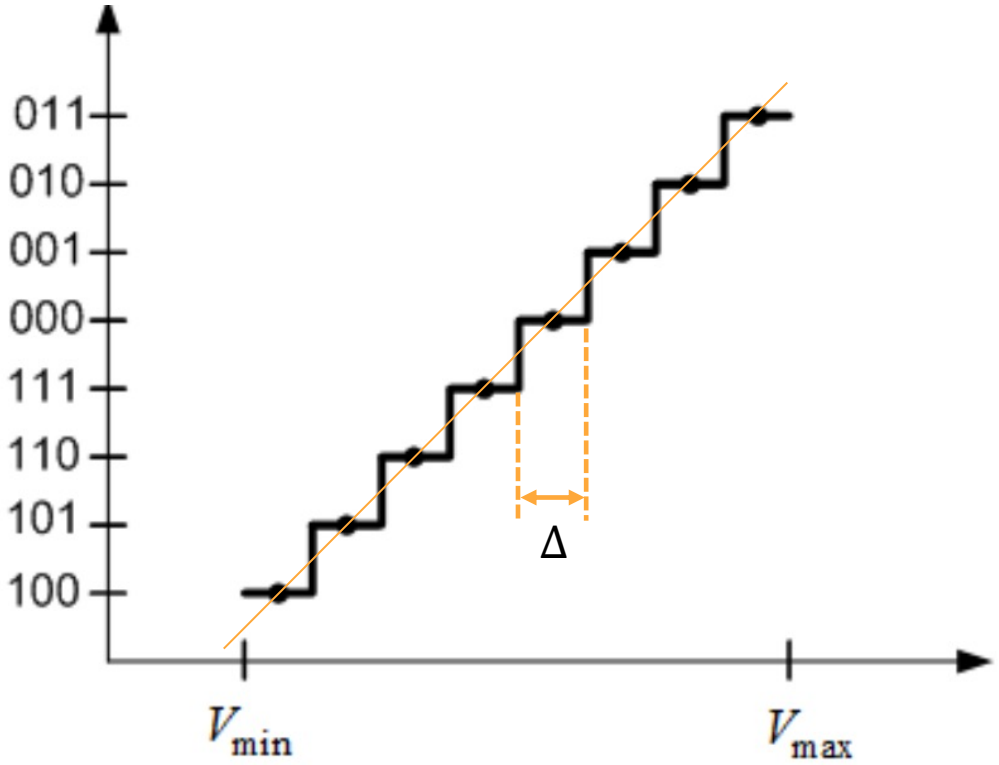
# Number Representation in Inference and Training




- Inference is based on **fixed-point** representations
- Training is based on **floating-point** representation to represent small error and gradient values

https://www.ibm.com/blogs/research/2018/12/8-bit-precision-training/

# Fixed-point Representation: Uniform Quantization




Uniform quantization

- Quantization step size:

$$\Delta = \frac{v_{max} - v_{min}}{2^B}$$

- Quantization error bounds:

$$|e| < \frac{\Delta}{2} = \frac{v_{max} - v_{min}}{2^{B+1}}$$

- Small bit precision $B$ creates large quantization error

# 2's Complement Number System

$X$ is $B$-bit 2's complement number with $\{x_{B-1}, x_{B-2}, \ldots, x_0\}$
Here, $x_b \in \{0,1\}$, and $-1 \leq X < 1$.
$x_{B-1}$ represents sign of the number (1: negative, 0 positive).

$$X = -x_{B-1} + \sum_{b=0}^{B-2} x_b 2^{b-B+1}$$

# Sign and Magnitude Number System

$X$ is $B$-bit sign and magnitude number with $\{x_{B-1}, x_{B-2}, \ldots, x_0\}$
Here, $x_b \in \{0,1\}$, and $-1 \leq X < 1$.
$x_{B-1}$ represents sign of the number.

$$X = (1 - 2x_{B-1}) * \sum_{b=0}^{B-2} x_b 2^{b-B+1}$$

# Floating-point Number Representation

IEEE 754 standard (32b Single precision)

| 31 | 30–23 | 22–0 |
|---|---|---|
| Sign | exponent | mantissa |

$0 \leq \text{mantissa} < 1$

Case1) Normal number (when exponent > 0 && exponent bits not all 1):

$$X = (-1)^S \times (1 + mantissa) \times 2^{exponent-127}$$

Case2) Denormal number (when exponent = 0):

$$X = (-1)^S \times (0 + mantissa) \times 2^{exponent-127}$$

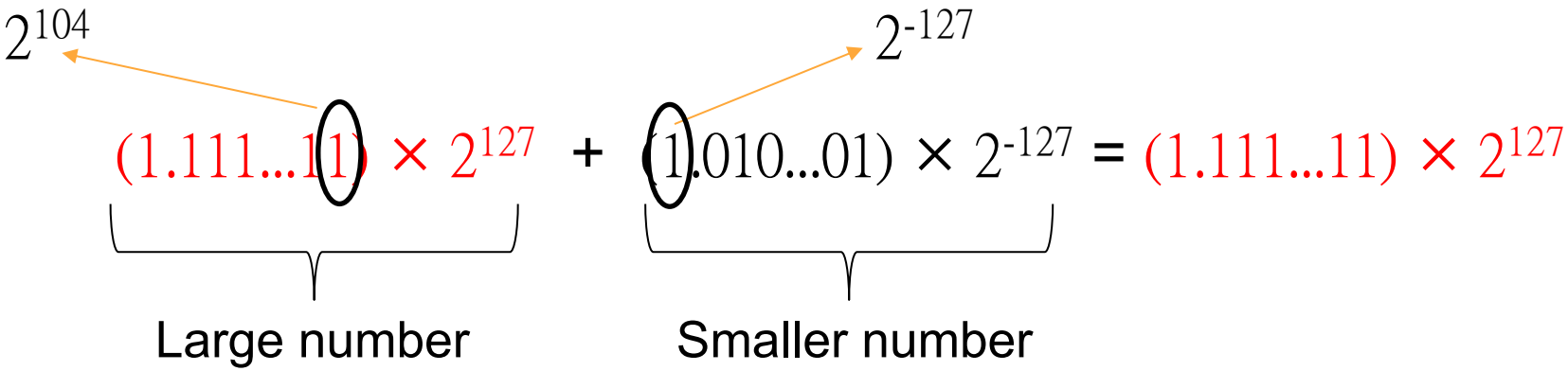
Case3) When exponent and mantissa = 0: $X = +0$ or -0

Case4) When all exponent bit = 1, and mantissa = 0: $X = +\infty$ or $-\infty$

Case5) When all exponent bit = 1, and mantissa ≠ 0: $X = \text{NaN}$

# Error Patterns in Floating-point Number

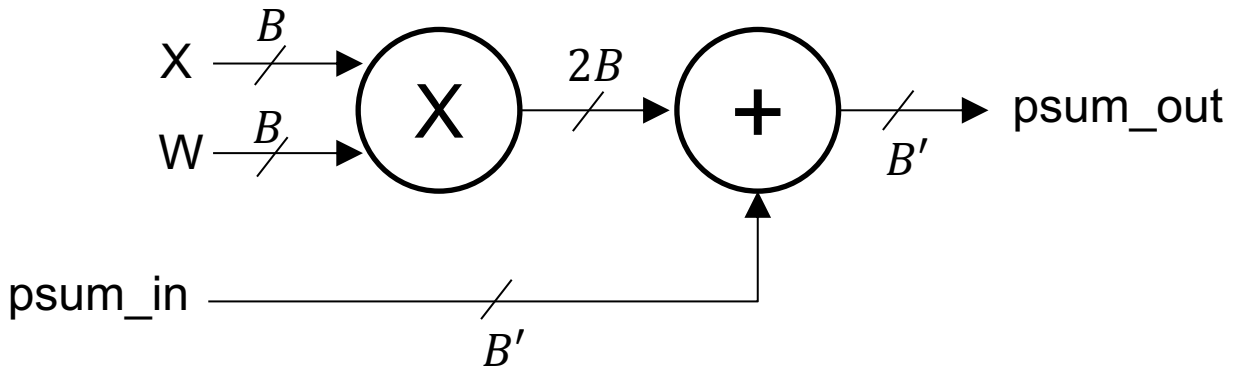
**Swamping error** example:




- When small number is added to large number, the small number is swamped

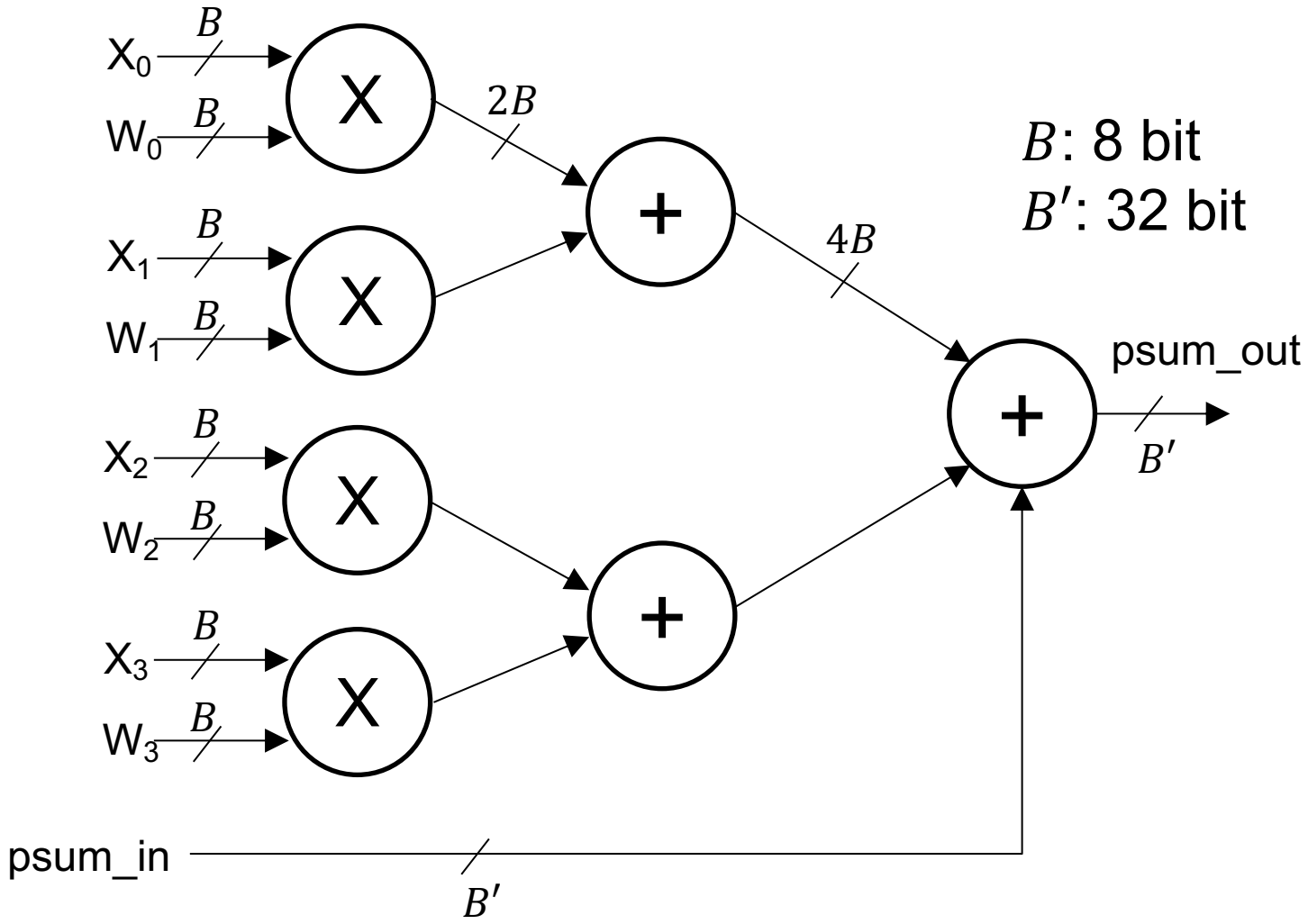
# Key Computing Kernel: Multiply-and-accumulation (MAC)

$$sum = \sum_{i=1}^{N} X_i W_i$$




- Multiply-and-accumulation (MAC) is a key computing kernel of DNN
- >7-80% of power consumption comes from MAC
- Multiplication and addition are designed together as a single unit, e.g., FMA
- MAC is considered as two operations (multiplication and addition)
- Output of multiplier needs twice more bit precision
- psum bit precision $B' = 2B + \log_2 N$

# Multi-input MAC



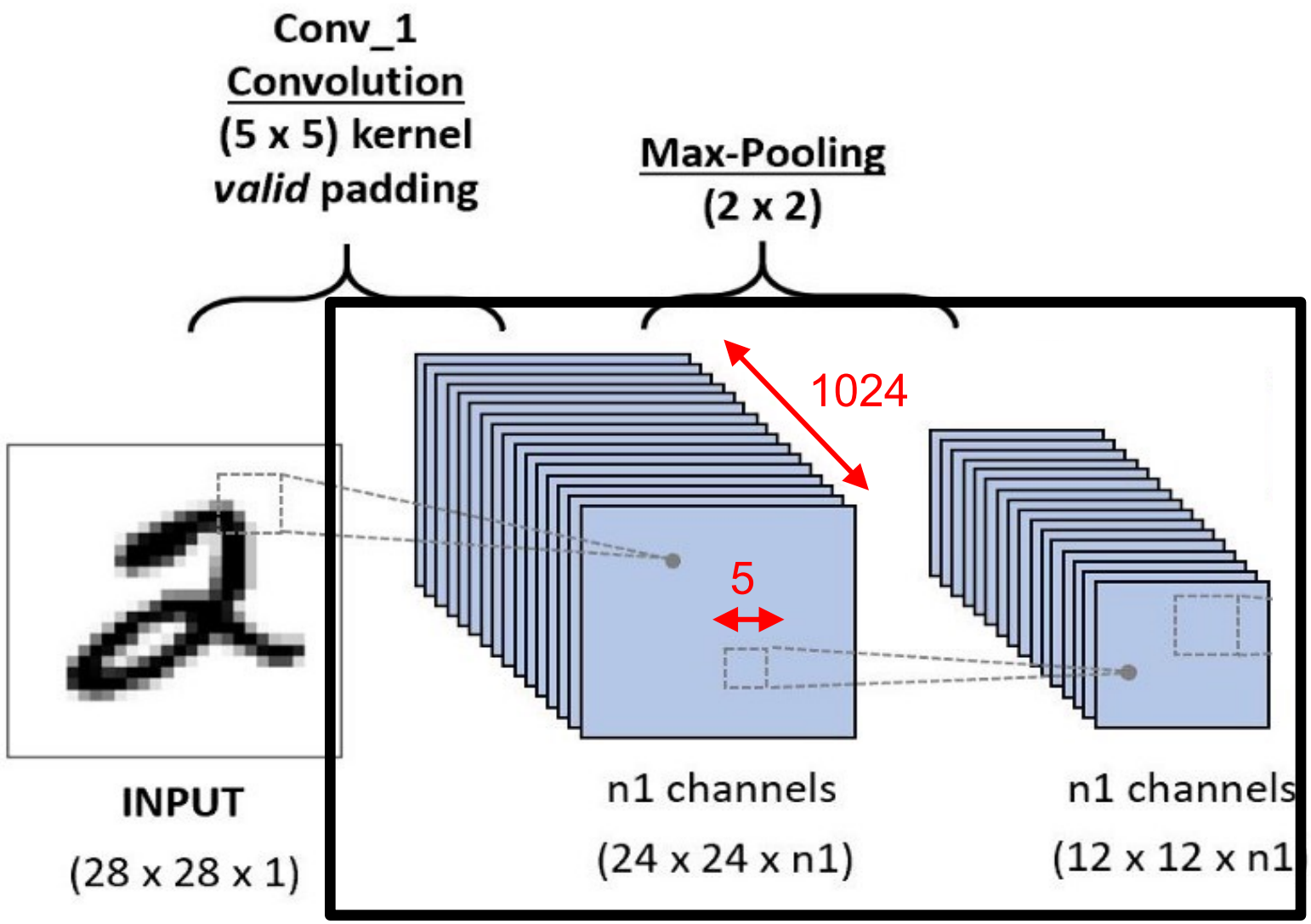

$B$: 8 bit
$B'$: 32 bit

| | Format | Instruction |
|---|---|---|
| INT4 | Two's complement | $R = \sum_8 A_i . Bi + C$ (A,B: 8x INT4; R,C: INT16) |
| INT2 | Weights: [-4, -1, 1, 4] Activations: unsigned int2 | $R = \sum_{16} A_i . Bi + C$ (A,B: 16x INT2; R,C: INT16) |

"A 7nm 4-Core AI Chip with 25.6TFLOPS Hybrid FP8 Training, 102.4TOPS INT4 Inference and Workload -Aware Throttling" ISSCC21

- Accumulation is taking more power in highly quantized system
- Multi-input MAC is employed

# Example of Bit Precision Assignment



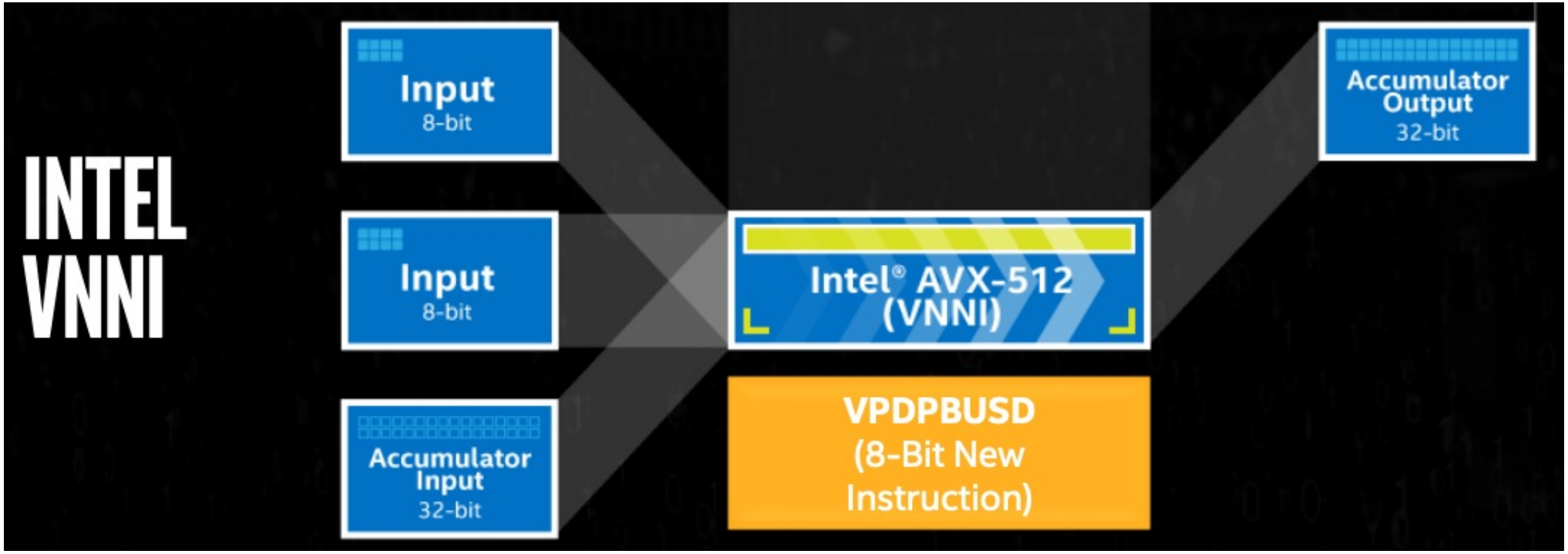

$$sum = \sum_{c=1}^{1024} \sum_{i \in 5 \times 5} X_{ci} W_{ci}$$

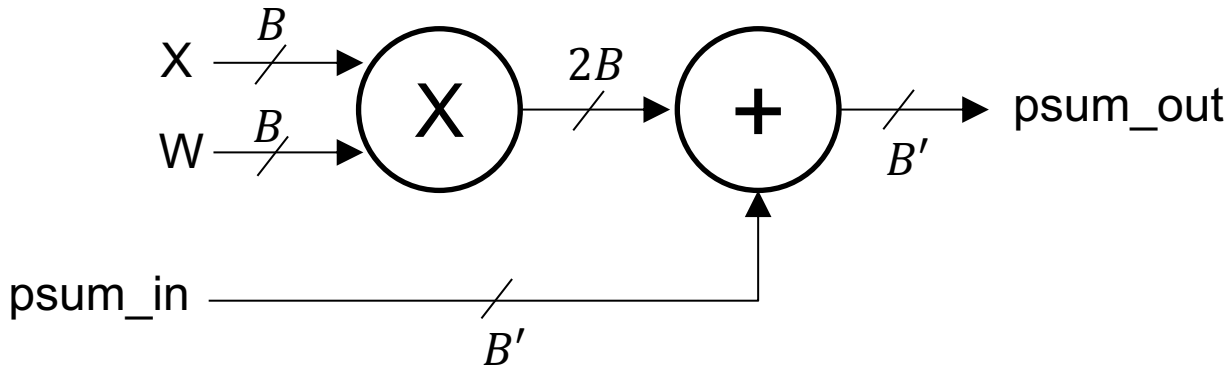
- psum bit precision

  $B' = 2B + \log_2(25 * 1024)$

- When $B = 8$, 31 bits needed for $B'$

# Example: Intel VNNI




$B$: 8 bit
$B'$: 32 bit