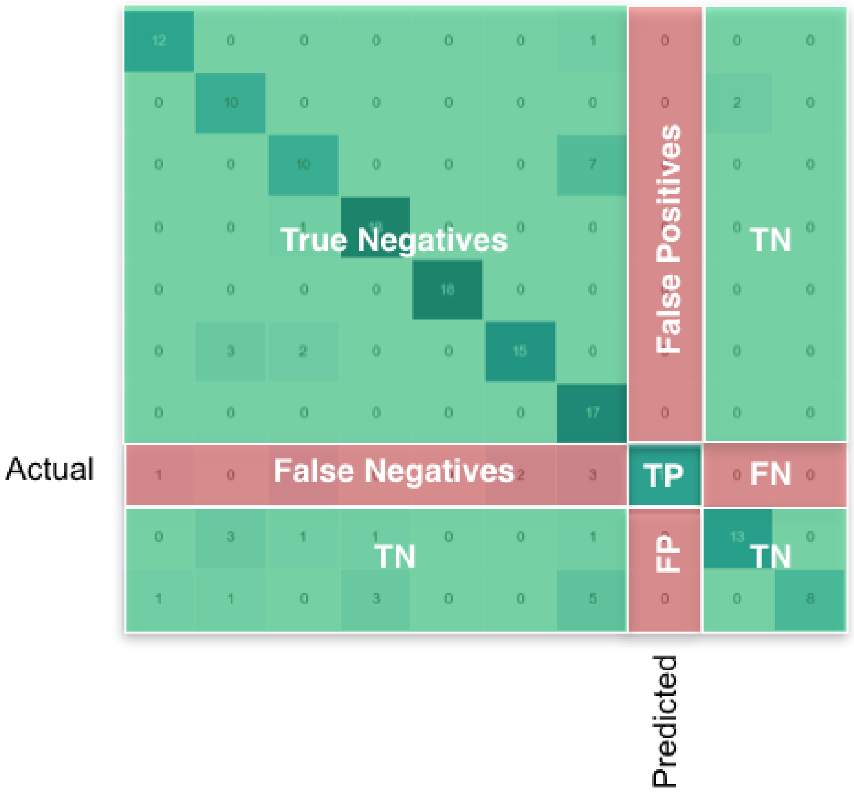
Quantification of Convolutional Neural Networks

Confusion Matrix

- The size of the confusion matrix is determined by the number of things we want to predict
- Example : MNIST classification (10 classes)



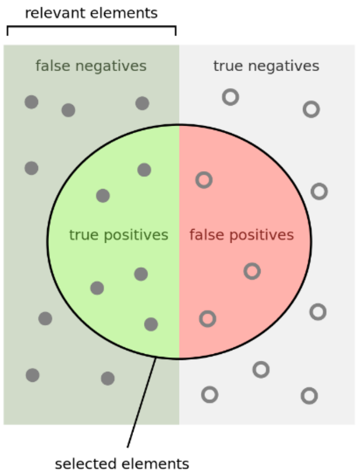
The diagonal are where the ML algorithm did right prediction

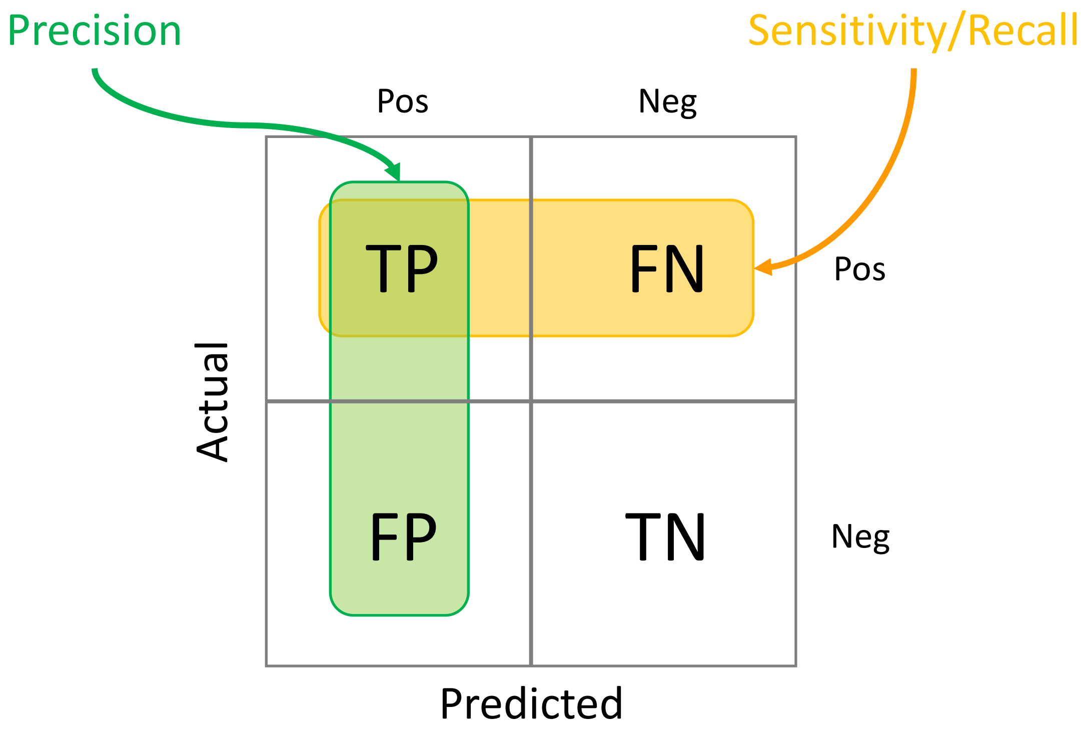
… and **everything else** is where the ML algorithm messed up

- https://androidkt.com/keras-confusion-matrix-in-tensorboard/

Precision, Sensitivity (Recall), Specificity, Accuracy

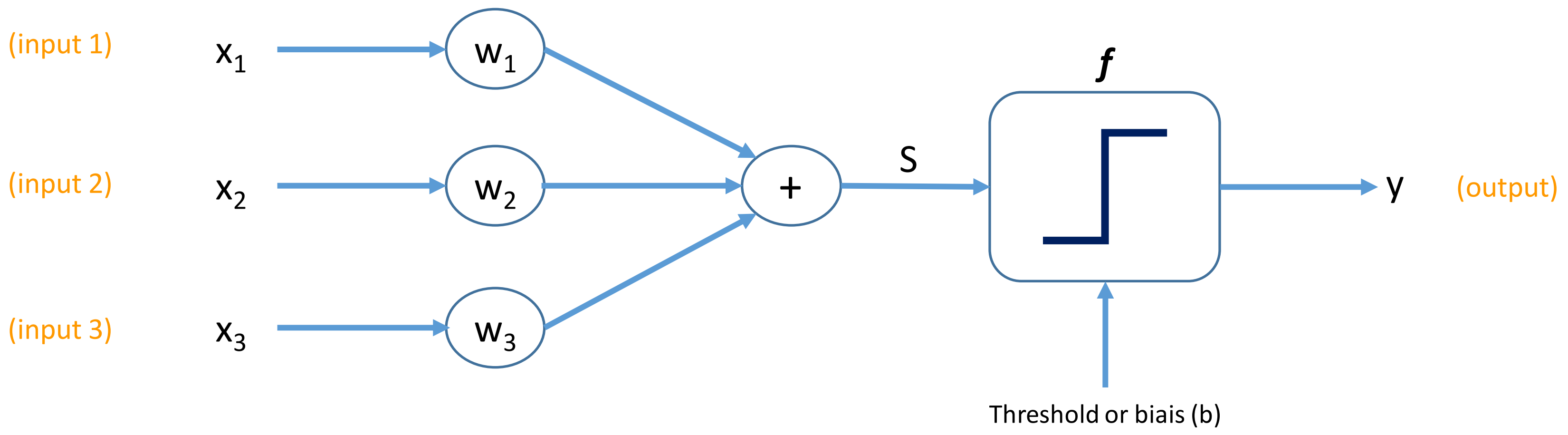
- $\textbf{\textit{Precision}} = \frac{TP}{TP+FP}$
- $\textbf{\textit{Sensitivity}} = \textit{Recall} = \frac{TP}{TP+FN}$
- $\textbf{\textit{Accuracy}} = \frac{TP+TN}{TP+TN+FP+FN}$
- $\textbf{\textit{Specificity}} = \frac{TN}{TN+FP}$
- $\textbf{\textit{Negative predictive Value}} = \frac{TN}{TN+FN}$





A Single Neuron

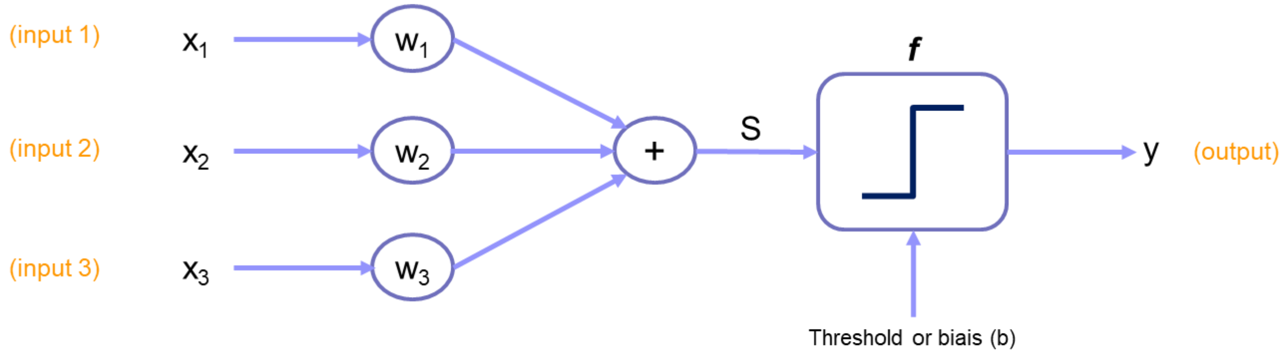
- The basic unit of computation in a neural network is the **neuron**, often called a **node** or **unit**.
- A neuron **receives input** from a source and **computes an output**.
- Each input has an associated **weight** (w), which is assigned on the basis of **its relative importance to other inputs**.
- There is a bias (b) that helps in controlling the value at which activation function will trigger.
- The node applies an activation function ***f*** to the weighted sum of its inputs as shown below:



Output of neuron = Y = $f\,(w_1.x_1 + w_2.x_2 + w_3.x_3 + b)$

Activation function

- The above network takes numerical inputs $\mathbf{X_1}$ and $\mathbf{X_2}$ and has weights **w1** and **w2** associated with those inputs.
- Additionally, there is another input with weight **b** (called the **Bias**) associated with it.
- The **function *f* can be non-linear** and is called the **Activation Function**.
- Particularly when used in neural network, the purpose of the activation function is to **introduce non-linearity into the output of a neuron**. This is important because **most real world data is non linear** and we **want neurons to *learn* these non linear representations**.



Output of neuron = Y = $\boldsymbol{f}\,(w_1.x_1 + w_2.x_2 + w_3.x_3 + b)$

Activation function

- Every **activation function** (or *non-linearity*) takes a single number and **performs a certain fixed mathematical operation on it**.

- There are **several activation functions** you may encounter in practice:
  - **Sigmoid or logistic function:** takes a real-valued input and squashes it to range between [0 ,1]

    $$\sigma(x) = \frac{1}{1+e^{-x}}$$

  - **tanh:** takes a real-valued input and squashes it to the range [-1, 1] (more efficient to compute than sigmoid)

    $$\tanh(x) = 2\sigma(2x) - 1$$

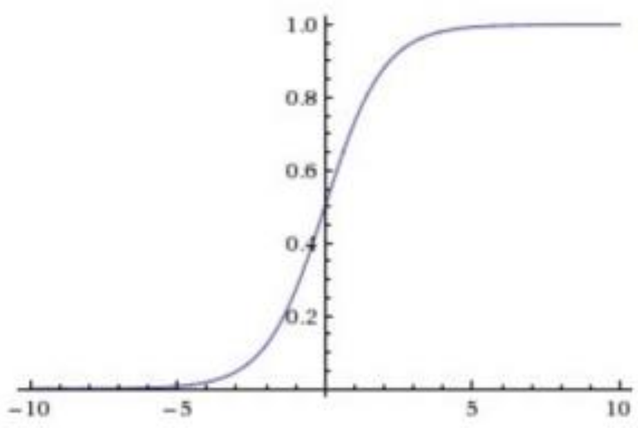
  - **ReLU**: ReLU stands for **Re**ctified **L**inear **U**nit. It takes a real-valued input and thresholds it at zero (replaces negative values with zero)

    $$f(x) = \max(0, x)$$

Activation function

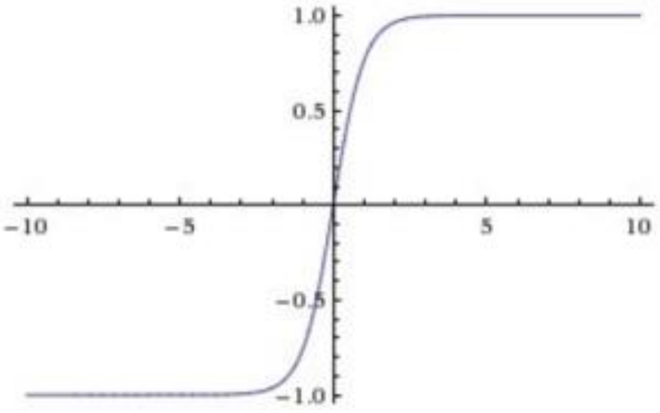
- The below figures show each of the above activation functions.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



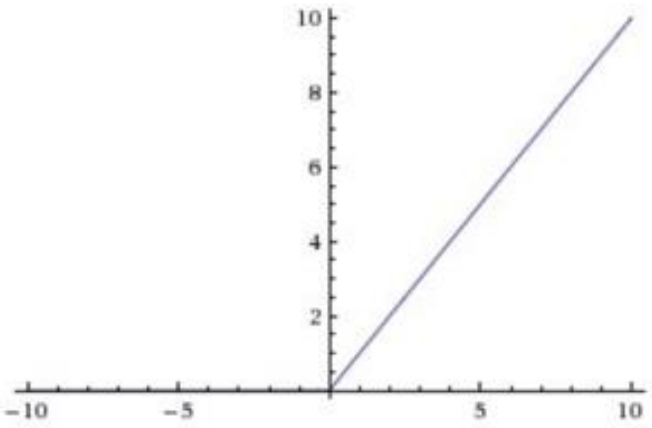
Sigmoid

$$\tanh(x) = 2\sigma(2x) - 1$$



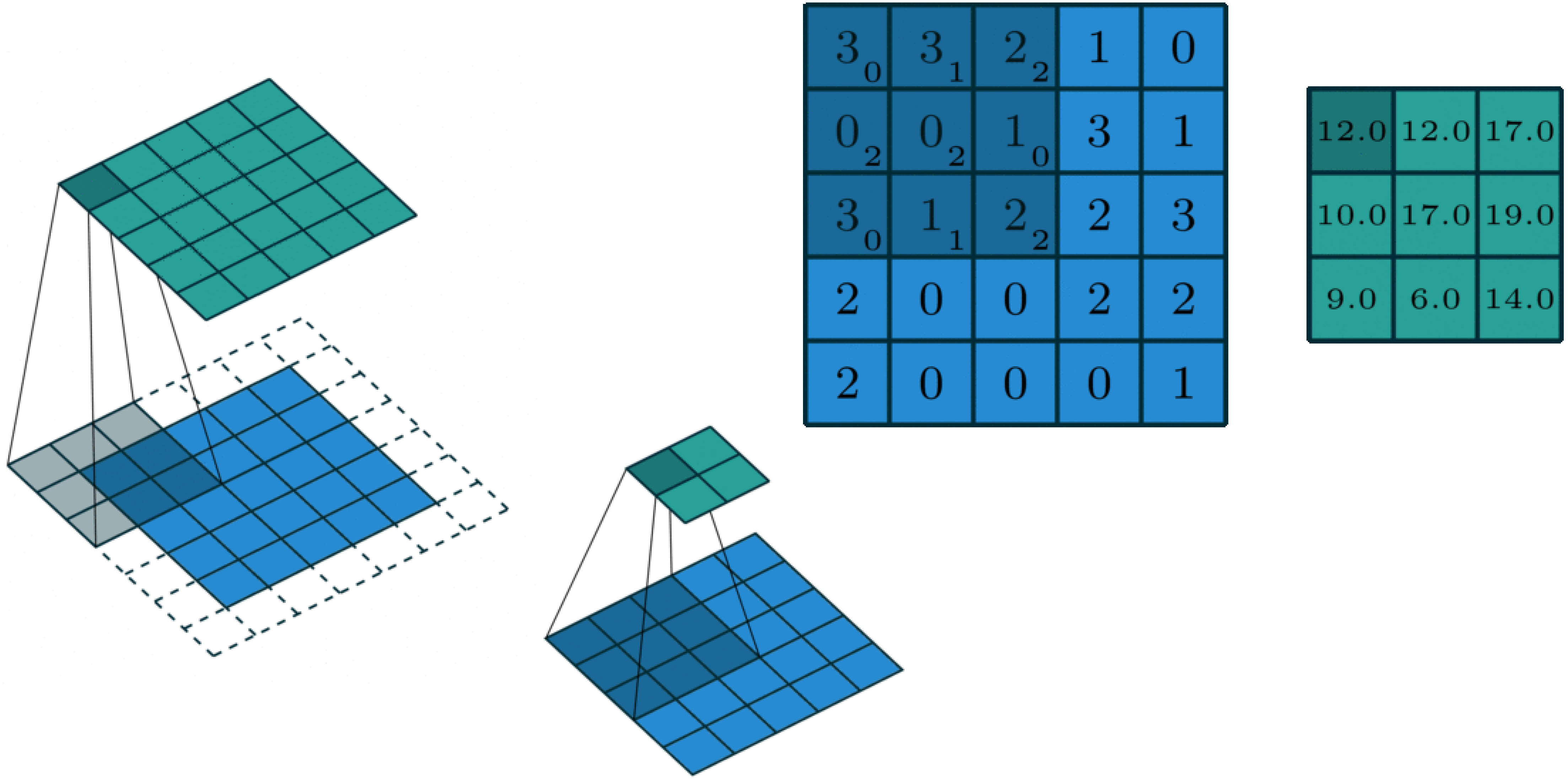
tanh

$$f(x) = \max(0, x)$$



ReLU

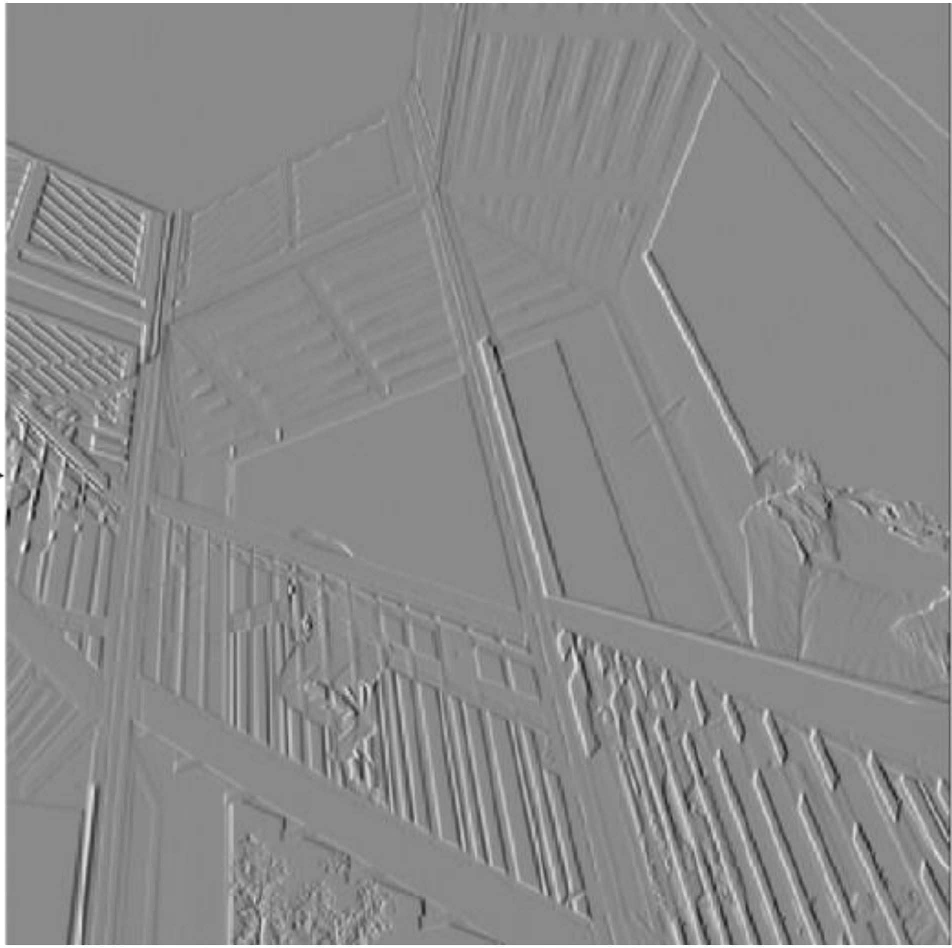
Convolutions



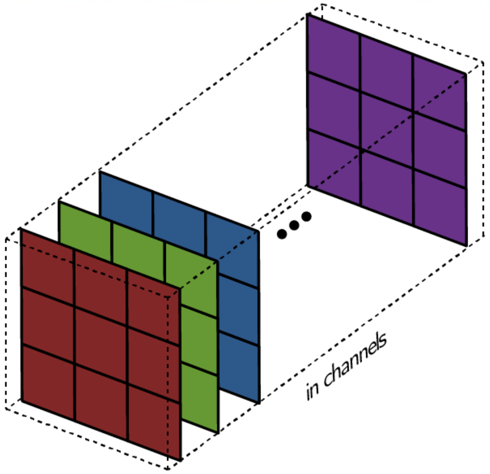
Example with an edge detector

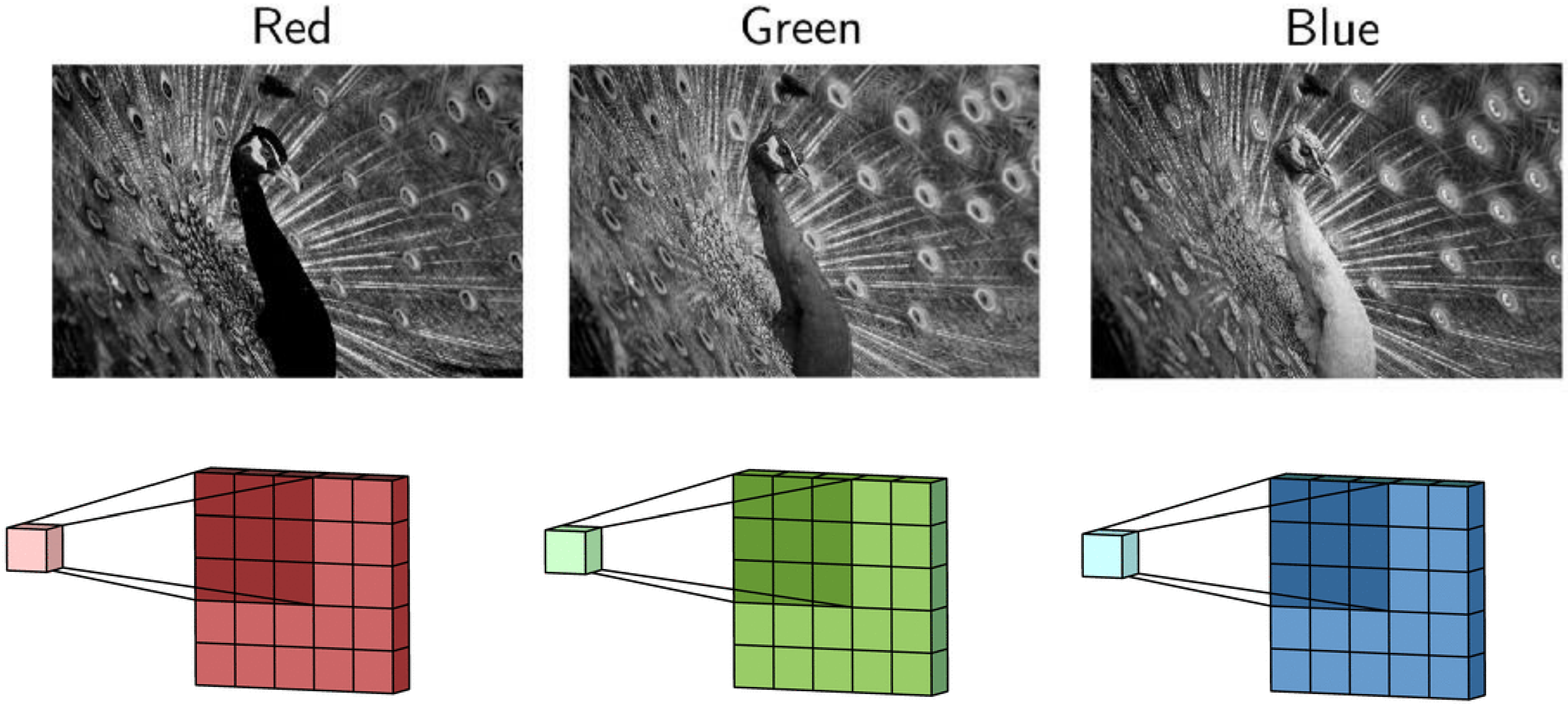
$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

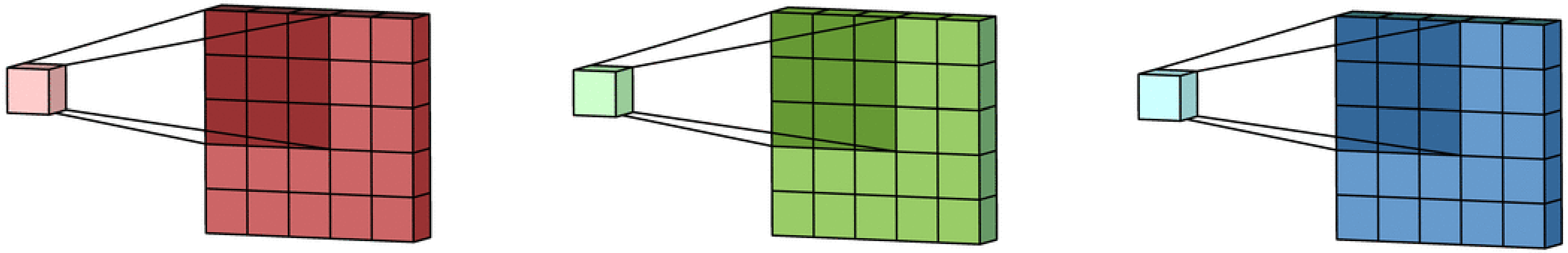
Horizontal Sobel kernel



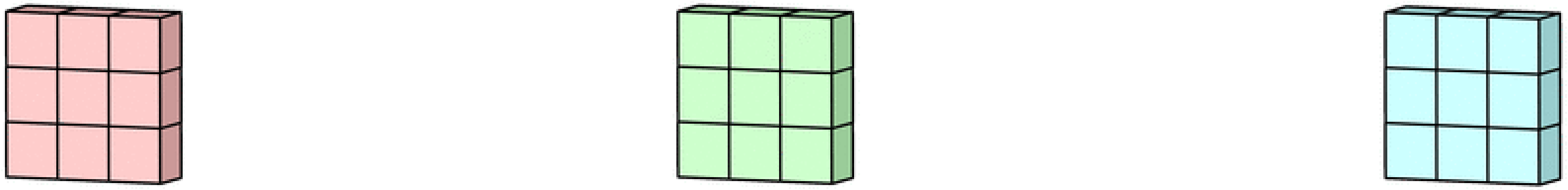
3 channels convolution



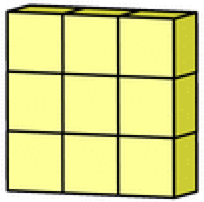




3 channels convolutions

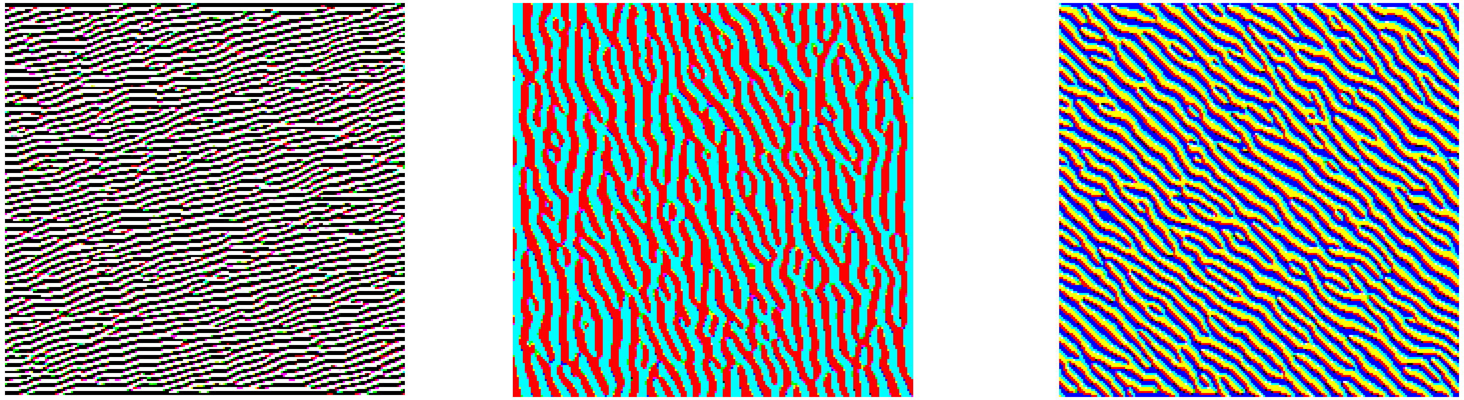


Bias addition

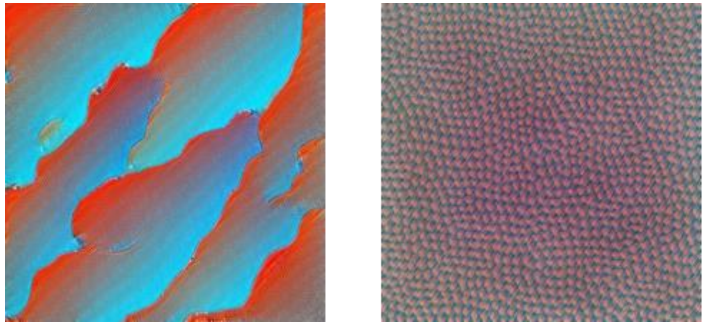


Each kernek generate an output channel by aggregation of the input channels

Visualization of Feature maps
on GoogLeNet

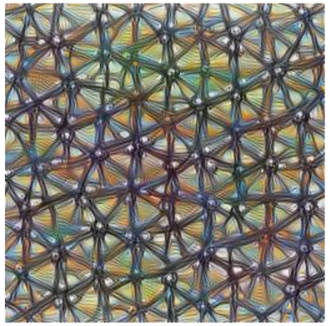


Feature visualization for 3 different channels from the 1st convolution layer of GoogLeNet[3].



Feature Visualization of channel 12 from the 2nd and 3rd convolutions[3]

mixed3a, channel 31



mixed4a, channel 11

mixed5a, channel 14

Feature visualization of channels from each of the major collections of convolution blocks, showing a progressive increase in complexity[3]

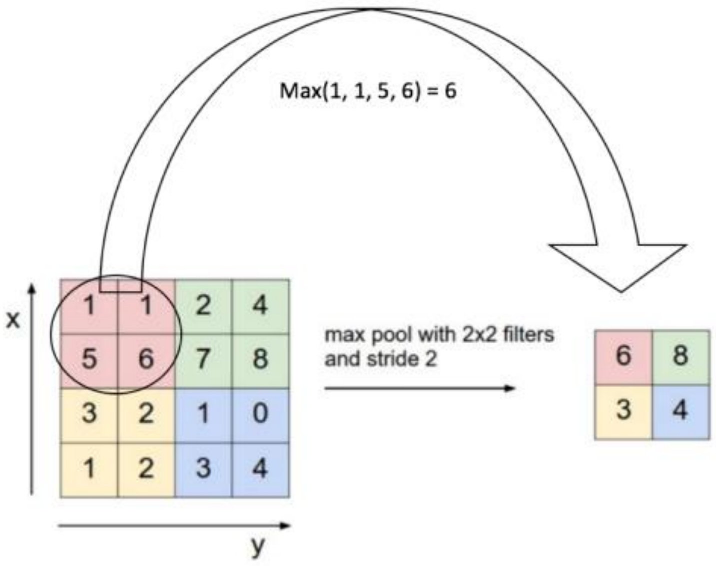
The Pooling Step

- **Spatial Pooling** (also called **subsampling** or **downsampling**) **reduces the dimensionality** of each feature map but **retains the most important information**.
- Spatial Pooling can be of different types: **Max**, **Average**, **Sum**, etc.

- In case of **Max Pooling**, we define a spatial neighborhood (for example, a 2×2 window) and **take the largest element from the rectified feature map within that window**.
- Instead of taking the largest element we could also take the average (Average Pooling) or sum of all elements in that window.
- In practice, **Max Pooling has been shown to work better**.

The Pooling Step

- The figure below shows an example of **Max Pooling operation** on a Rectified Feature map (obtained after convolution + ReLU operation) by **using a 2×2 window**.

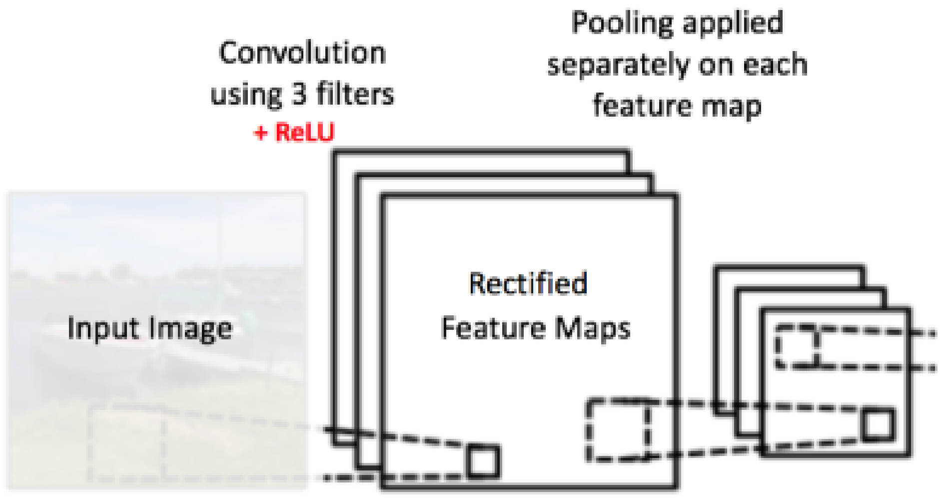


Rectified Feature Map

- We slide our 2 x 2 window by 2 cells ('stride') and **take the maximum value in each region**.
- As shown, **this reduces the dimensionality of our feature map**.

The Pooling Step

- In the network shown below, **pooling operation is applied separately to each feature map**.
- Due to this, we get therefore **3 output maps from 3 input maps**.



IA on the Edge : Number representation

Floatting point: Used for training (CPU/GPU) for more precision

Need a FPU

Fixed point: Used for fine-tuning and inference on target

Stored and computed as integers

⇒ Need for a conversion from floating-point to fixed-point:

⇒ Determine **a scale** factor so that a floating-point can be represented as an integer multiplied by a scale factor

The scale is a factor of 2 => computed as shifts

Has to be choosen to represent the whole range of values while avoiding and risk of data overflow

Pierre-Emmanuel Novac, Ghouthi Boukli Hacene, Alain Pegatoquet, Benoît Miramond, Vincent Gripon. Quantization and Deployment of Deep Neural Networks on Microcontrollers. *Sensors*, MDPI, 2021

IA on the Edge : Deployment on MCU

After the network has been trained and quantized it is deployed:

- **Export the weights** of the DNN and encode them into a format suitable for on-target inference
- **Generate the inference program** according to the topology of the DNN
- **Compile** the inference program
- **Upload** the program with weights onto the MCU’s ROM

Pierre-Emmanuel Novac, Ghouthi Boukli Hacene, Alain Pegatoquet, Benoît Miramond, Vincent Gripon. Quantization and Deployment of Deep Neural Networks on Microcontrollers. *Sensors*, MDPI, 2021

IA on the Edge : Deployment on MCU

MicroAI framework

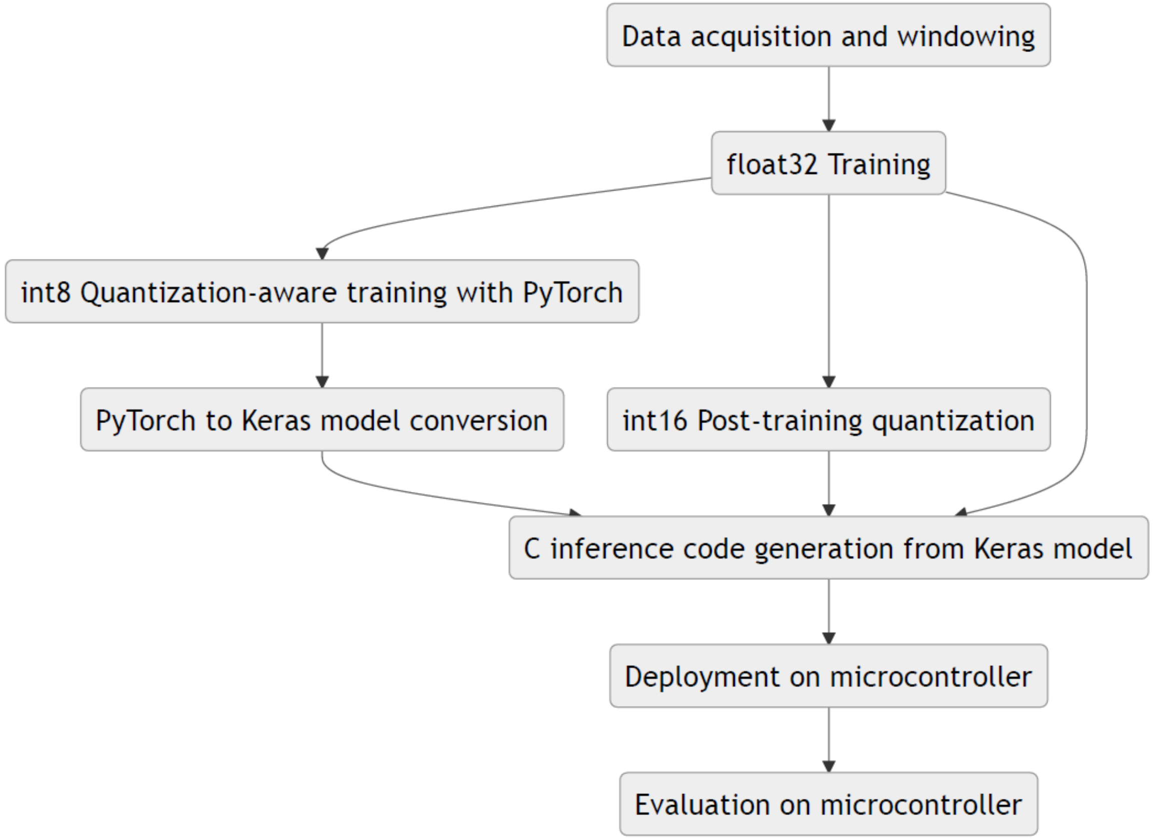
Open-source
Support CNN with non-sequential topologies
Easy to modify and extend

Built in two parts:
1. A neural network training code that relies on Keras or PyTorch
2. A conversion tool (KerasCNN2C) that takes a trained Keras model and produces a portable C code for the inference

Pierre-Emmanuel Novac, Ghouthi Boukli Hacene, Alain Pegatoquet, Benoît Miramond, Vincent Gripon. Quantization and Deployment of Deep Neural Networks on Microcontrollers. *Sensors*, MDPI, 2021

IA on the Edge : Deployment on MCU

MicroAI General Flow

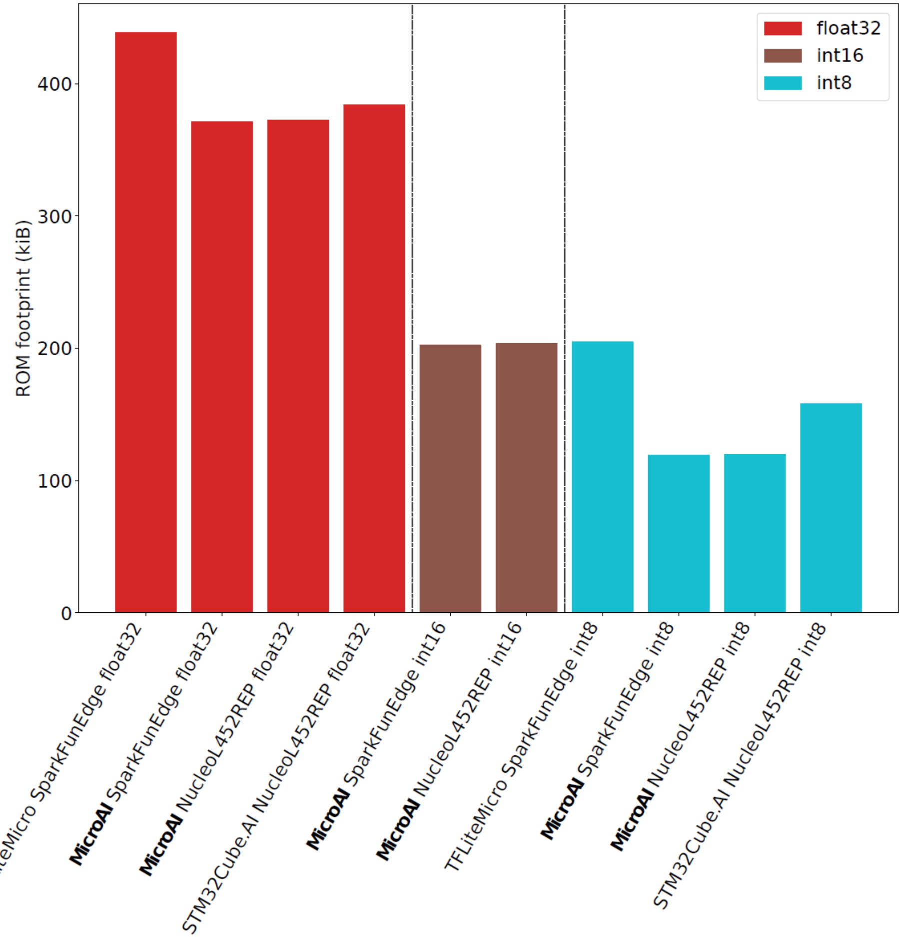


Pierre-Emmanuel Novac, Ghouthi Boukli Hacene, Alain Pegatoquet, Benoît Miramond, Vincent Gripon. Quantization and Deployment of Deep Neural Networks on Microcontrollers. *Sensors*, MDPI, 2021

Board	**Nucleo-L452RE-P**	**SparkFun Edge**
MCU	STM32L452RE	Ambiq Apollo3
Core	Cortex-M4F	Cortex-M4F
Max Clock	80 MHz	48 MHz (96 MHz “Burst Mode”)
RAM	128 KiB	384 KiB
Flash	512 KiB	1024 KiB
CoreMark/MHz	3.42	2.479
Run current @3.3 V, 48 MHz	4.80 mA	0.82 mA *

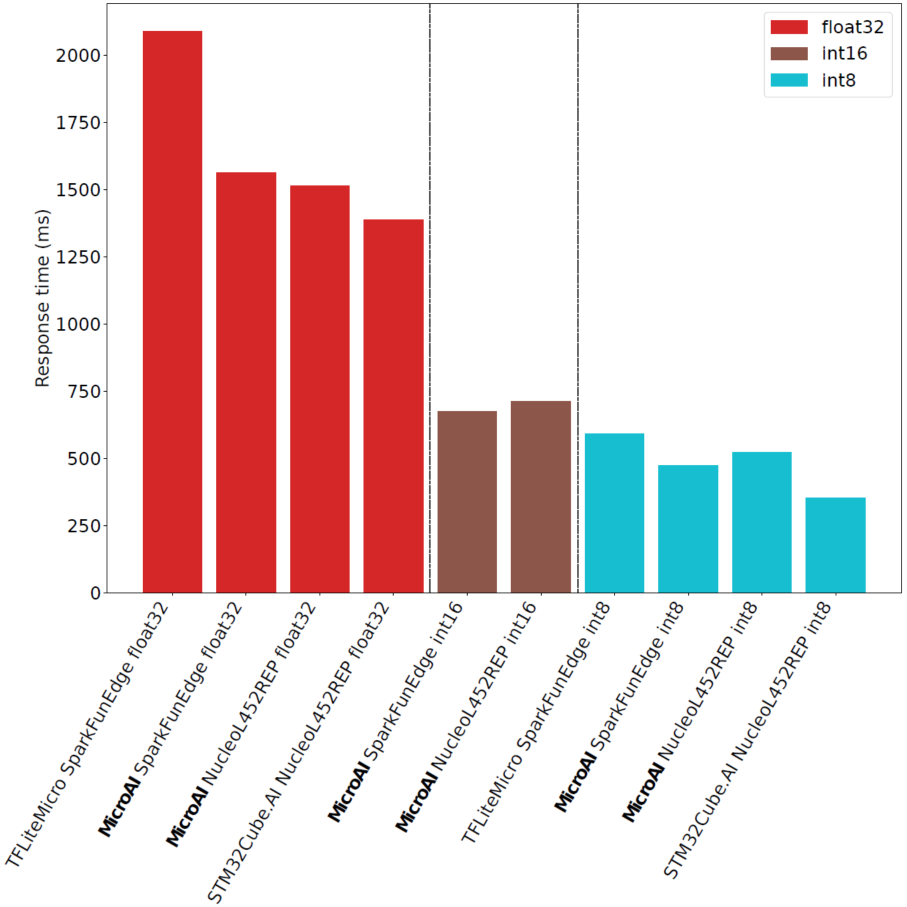
* After removing peripherals (Mic1&2, accelerometer...)

Deployment of deep neural networks on microcontrollers



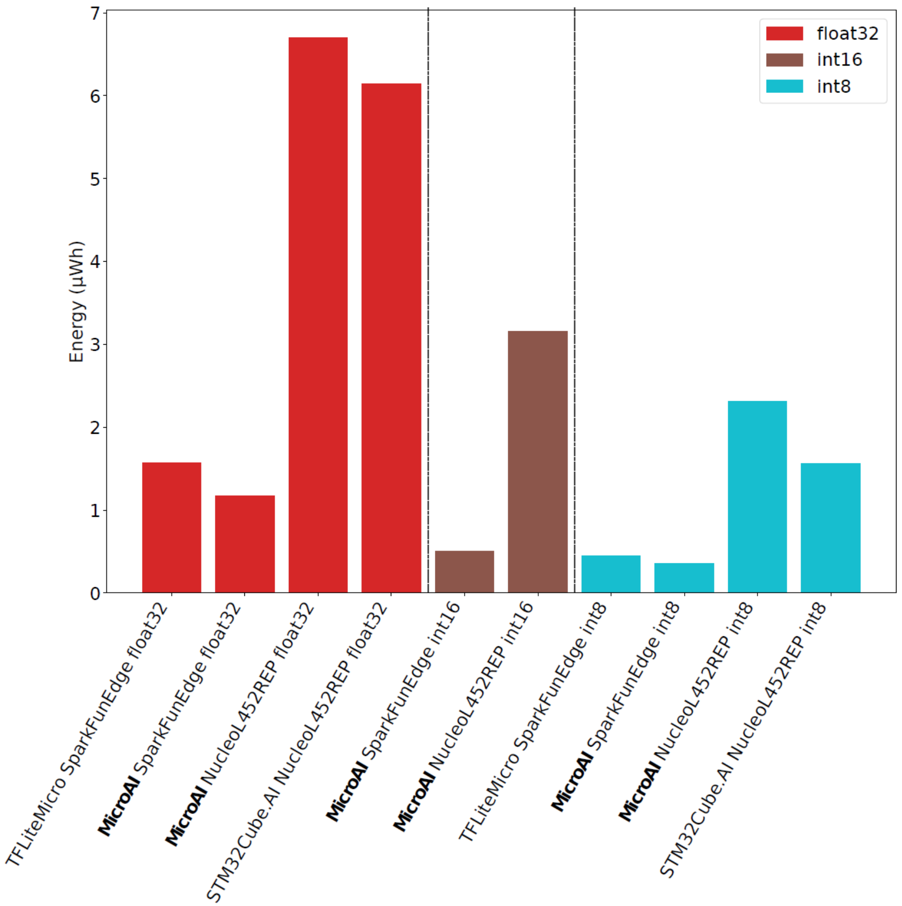
- MicroAI has the lowest memory overhead in all situations.

Deployment of deep neural networks on microcontrollers



- MicroAI is faster than TFLite Micro
- MicroAI is slightly slower than STM32Cube.AI

Deployment of deep neural networks on microcontrollers



- The Ambiq Apollo3 MCU brings a much better efficiency over the STM32L452RE

Quantization and deployment of deep neural networks on microcontrollers

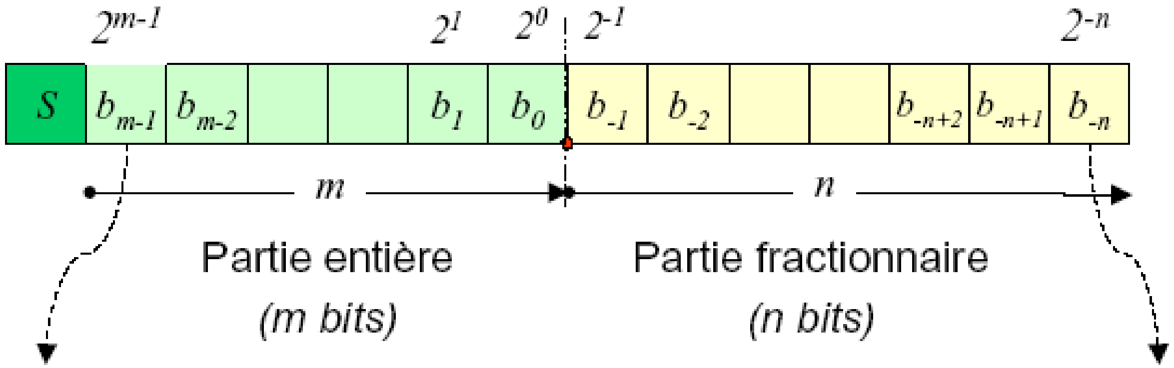
- int16: lower power consumption and ROM footprint than float32 without loss in accuracy
- int8: reduces them even further but with noticeable loss in accuracy.
  - Better quantification may help mitigate the loss
- MicroAI has a lower overhead on the ROM footprint than competitive solutions
- MicroAI inference time is similar to competitive solutions

Both the hardware and the software must be optimized together to achieve the lowest power consumption.

Quantification des nombres flottants

a. Notation à virgule fixe

$$110.111 = 1.2^2 + 1.2^1 + 0.2^0 + 1.2^{-1} + 1.2^{-2} + 1.2^{-3}$$



Domaine de définition
du codage :

$$D = [-2^m, 2^m - 2^{-n}]$$

Précision du codage
Pas de quantification :

$$q = 2^{-n}$$

Résolution et dynamique

- Résolution : différence entre 2 nombres consécutifs
- Dynamique : différence entre le plus petit nombre et le plus grand

N= $\boxed{2^{m-1} \qquad\qquad\qquad 2^{0}}$

MSB LSB

Résolution : 1

Dynamique : 2^{m-1}

N= $\boxed{2^{m-1} \qquad 2^{0}\ 2^{-1}\ 2^{-n}}$

MSB LSB

Résolution : 2^{-n}

Dynamique :

$$2^{m} - 2^{-n}$$

Gamme des nombres représentables en virgule fixe

Format	Limite négative	Limite positive	Résolution
Q1.31	-1	0,99999999…	$4,656612.10^{E}-10$
Q1.15	-1	0,99996948…	0,000030517
…			
Q9.7	-256	255,9921	0,0078125
Q10.6	-512	511,9843	0,015625
Q11.5	-1024	1023,96875	0,03125
Q12.4	-2048	2047,9375	0,0625
Q13.3	-4096	4095,875	0,125
Q14.2	-8192	8191,75	0,25
Q15.1	-16384	16383,5	0,5
Q16.0	-32768	32767	1

Virgule Fixe

- Avantages
  - Résolution constante
  - Arithmétique simple
    - Facilite l’addition : semblables aux entiers
    - Multiplication : Nécessite des décalages supplémentaire
- Utilisation
  - Dans des circuits spécifiques ou DSP
  - Peu dans les ordinateurs généralistes

b. Notation en virgule flottante

- Nombre flottant N en binaire :
  - Un bit de signe s
  - Un exposant e
  - Une mantisse m

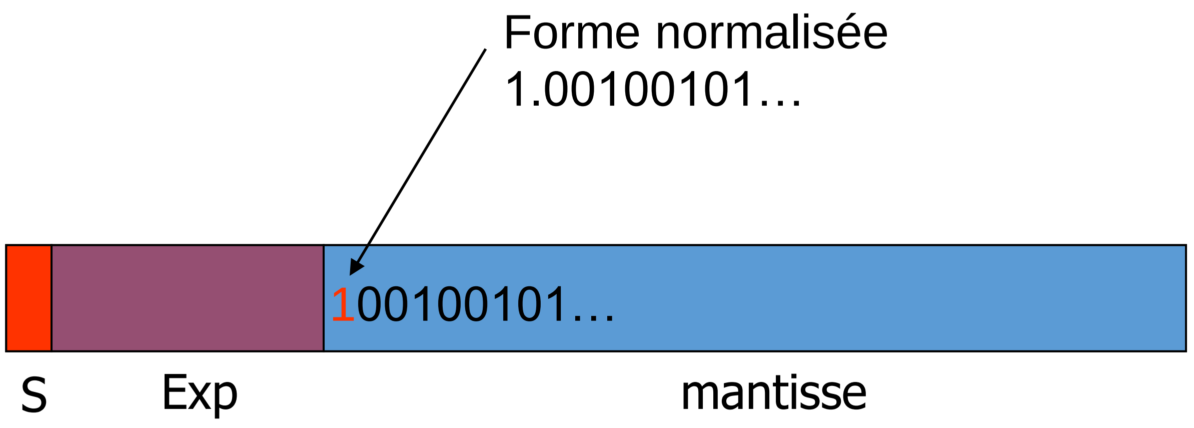
$$N = (-1)^s \times m.2^e$$

- Représentations équivalentes

$$0.0000101011.2^0$$

$$0.000000101011.2^2$$

$$1.01011.2^{-5}$$

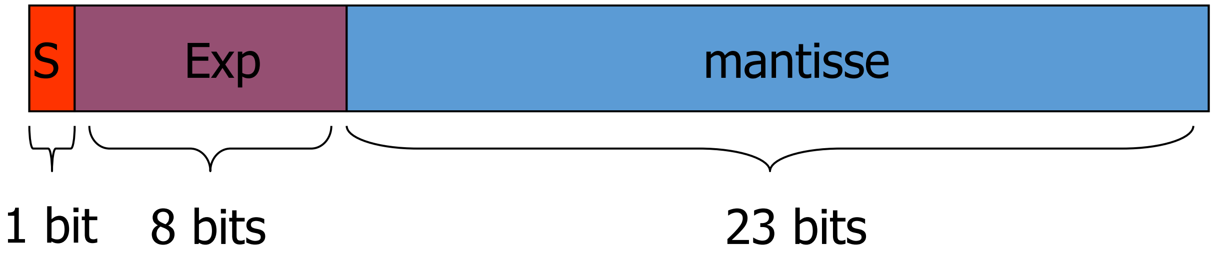


- Normalisation:
  - Le chiffre le plus significatif (non nul) est placé à l’extrême gauche de la mantisse

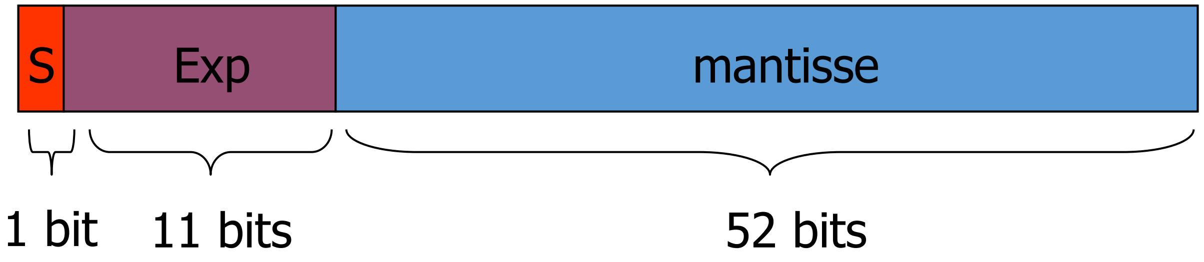
La norme IEEE 754

- Objectifs de la norme
  - Représentation des nombres
  - Procédures d’arrondis
  - Précision
  - Traitement des exceptions
- Principe
  - Toujours 1 avant la virgule (ce bit n’est pas codé)
  - 1<mantisse<2

Précision simple sur 32 bits (10**-38 à 10**38)

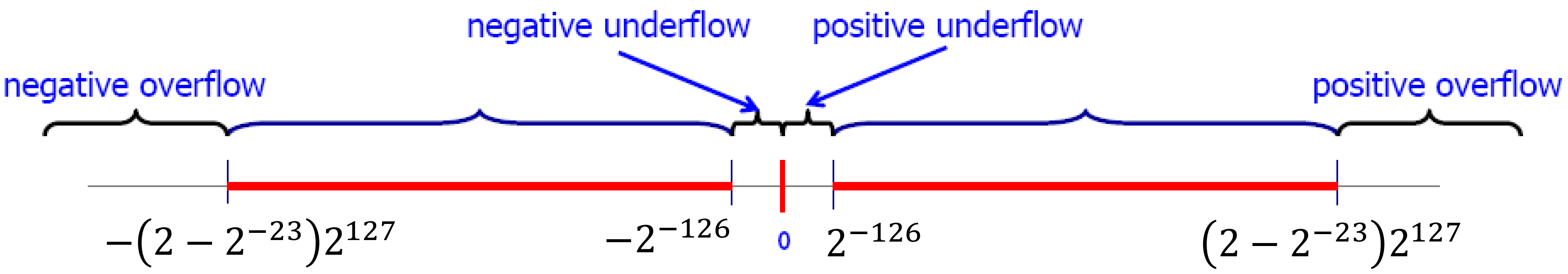


Précision double sur 64 bits (10**-308 à 10**308)



- Codage de l’exposant
  - L’exposant n’est pas représenté en complément à 2. Il est biaisé
  - Biais = 127
  - exp codé = biais + exp réel
  - exp réel = exp codé - biais

Précision



- Pour un nombre total de bits constant, il faut faire un compromis entre rang et précision
  - Si on augmente la taille de l'exposant le rang augmente mais la précision diminue
- Ecart non constant entre les nombres
  - Résolution :

$$\Delta N = 2^{-23}.2^{E-126}$$

- Par contre, la précision relative est constante

$$P = \frac{2^{-23}}{1,F} \leq 2^{-23} \approx 1,2.10^{-7}$$

Valeurs particulières

Normalisé

+ -	0<Exp<max	Configuration quelconque de bits

Dénormalisé

+ -	0	Configuration quelconque de bits non nulle

zéro

+ -	0	0

+ infini ou -infini

+ -	111..1	0

NaN: not a number

+ -	111..1	Configuration quelconque de bits

Les nombres dénormalisés

- L’exposant est nul
- La mantisse n’est plus normalisée
  - Il n’y a plus de bit implicite à 1 : $0<m<1$
  - Codée sur 23 ou 52 bits

- Plus petit nombre dénormalisé représentable
  - 23 bits à 0 + dernier bit à 1
  - Mantisse 2^-23, Exposant 2^-126 soit 2^-150
- Plus grand nombre dénormalisé représentable
  - 0.9999999.2^(-127)

Précision

- ΔNd :Distance dénormalisée entre 2 nombres
- Pd : Précision relative

$$\Delta N_d = 2^{-23}.2^{-126} = 2^{-149}$$

$$P_d = \frac{2^{-23}}{0,F}$$

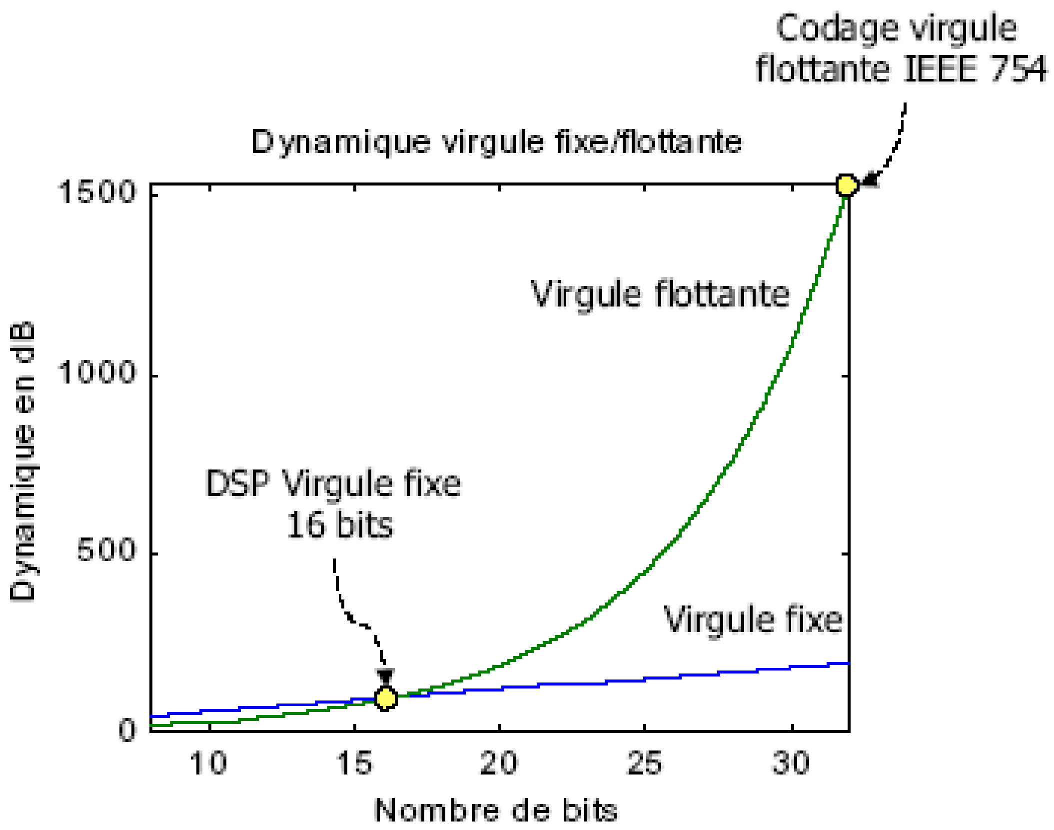
- Précision absolue constante comme dans le cas des nombres en virgule fixe
- Précision relative inférieure à celle des flottants normalisés (cas des plus petits nombres)

Comparaison
Simple / Double précision

Caractéristique	Simple	Double
Taille totale	32	64
Taille exposant	8 [-126, +127]	11 [-1022, + 1023]
Taille mantisse	23	52
Codage mantisse	Signe & Grandeur	Signe & Grandeur
Codage exposant	Par Excédant 127	Excédant 1023
Plus petit positif normalisé	$1{,}0.2^{E}$-126	$1{,}0.2^{E}$-1022
Plus petit positif dénormalisé	(2E-23).2E-126	$(2^{E}$-52).2^{E}-1022
Plus grand positif normalisé	$(2-2^{E}$-23).2E+127	$(2-2^{E}$-52).2E+1023
Plus grand positif dénormalisé	$(1-2^{E}$-23).2E+127	$(1-2^{E}$-52).2E+1023

Comparaison Virgule fixe/Virgule flottante

- Niveau de la dynamique



Implantations matérielles

Problèmes relatifs aux nombres flottants

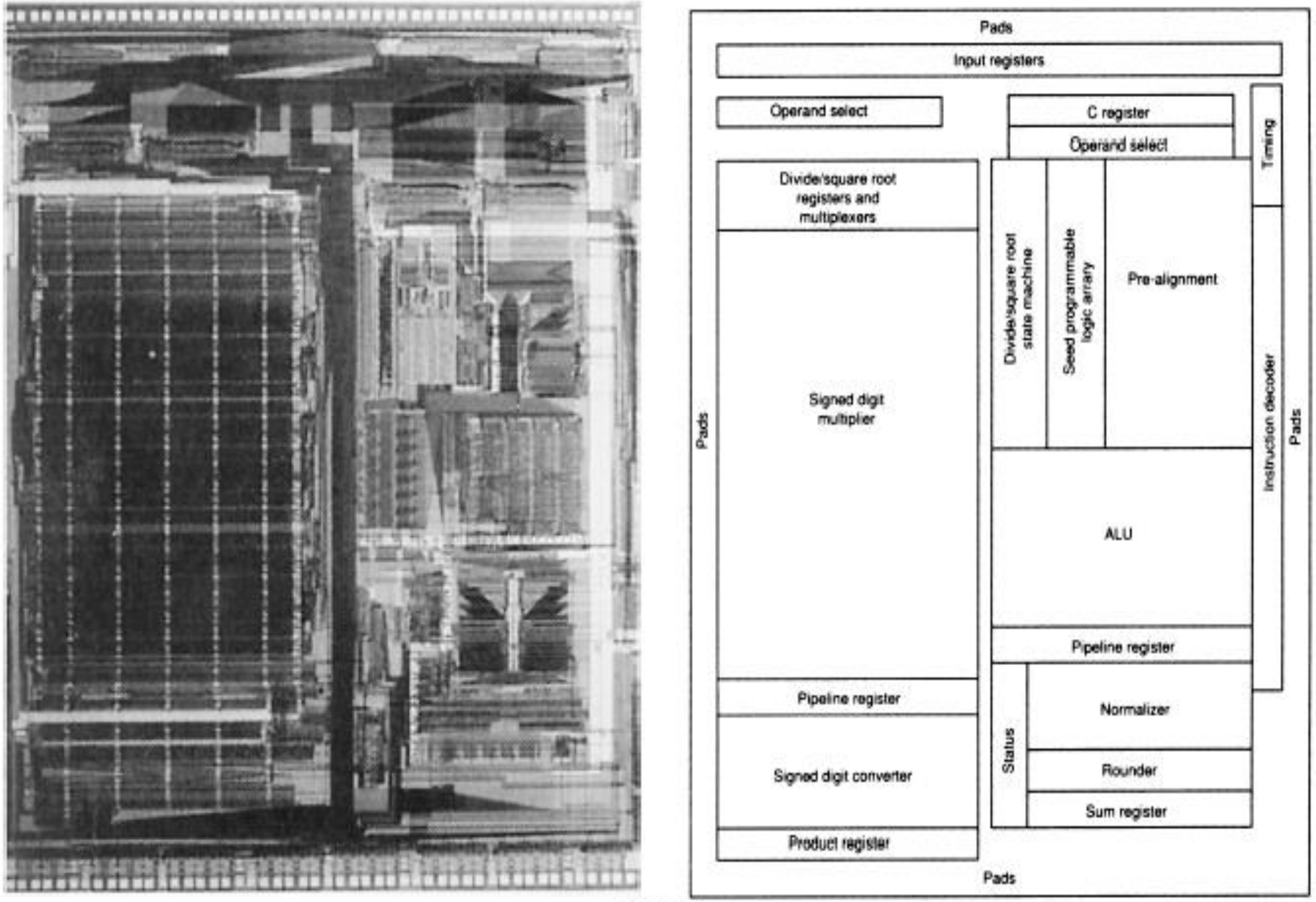
- **Le problème du missile patriote**
- Durant la 1ère guerre du Golfe (1991), un missile patriote à loupé l'interception d'un missile scud irakien
- Mauvais calcul du temps de vol du missile patriote
- Le temps estimé en dizième de seconde a été converti en seconde en multipliant par 1/10.
- L'opération a été tronquée sur 24bits (représentation en virgule fixe) et l'erreur s'est propagée induisant un retard de 0.34s

Explosion d'ARIANE 5

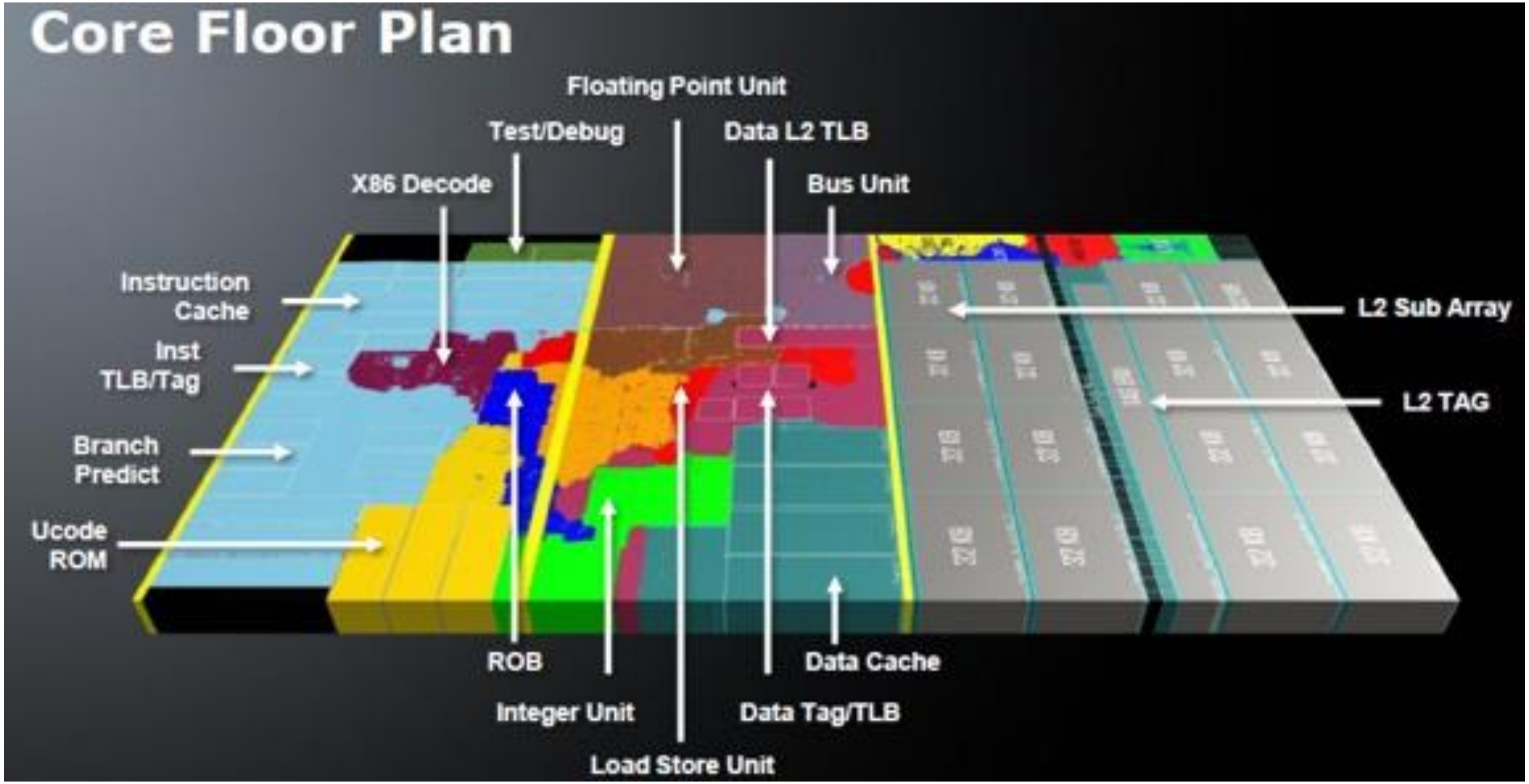
- Problème dans le logiciel de lancement
- Conversion d'un nombre flottant sur 64 bits en un entier signé codé sur 16 bits
- La valeur à convertir était plus grande que 32767 !!

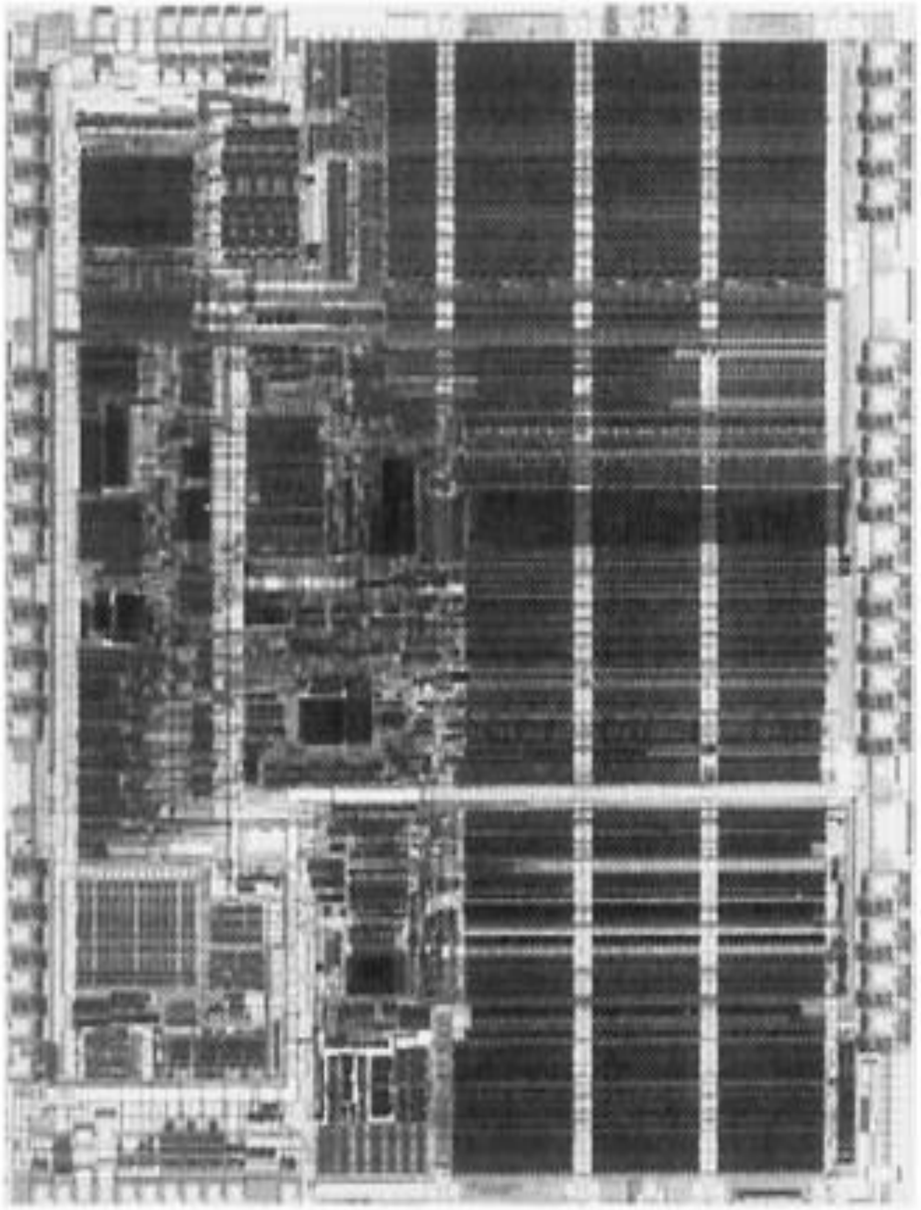
Implémentation

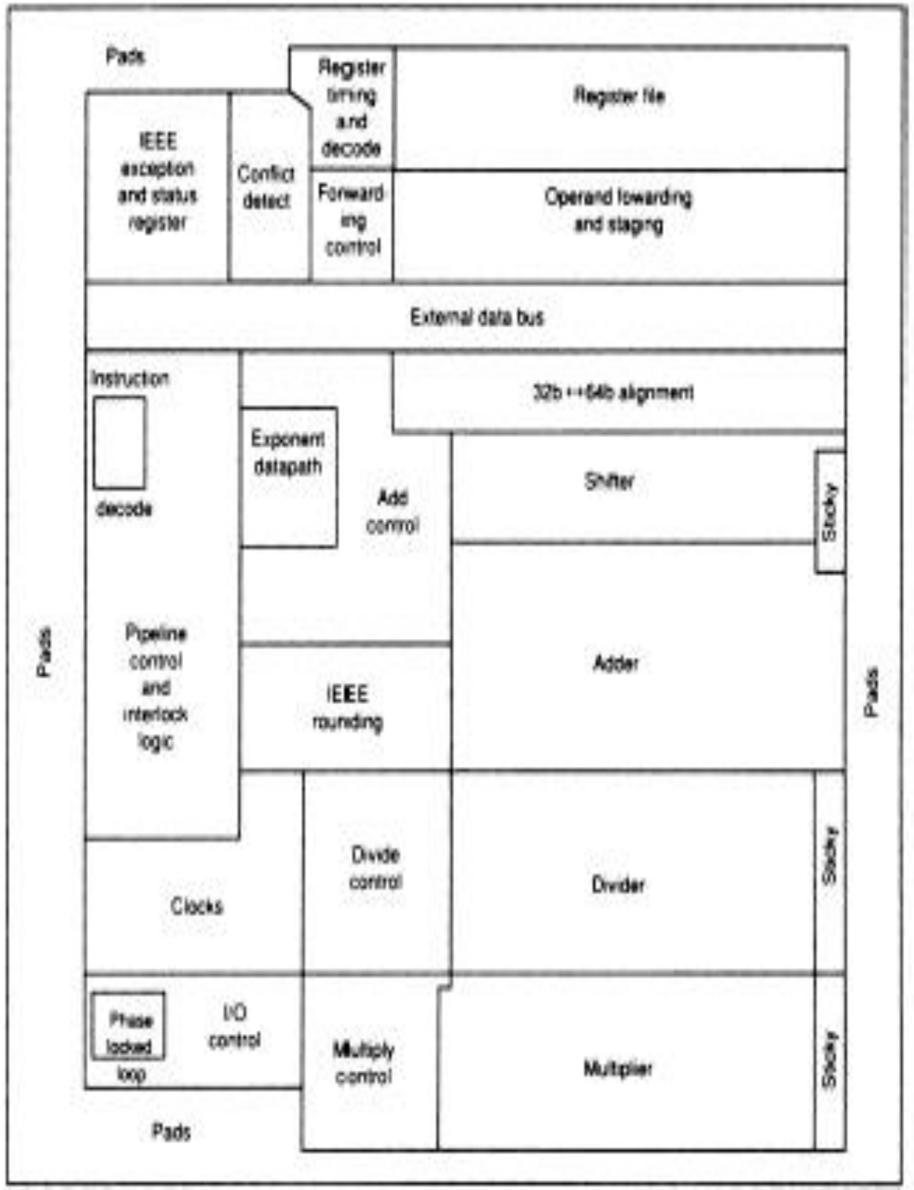
- Virgule fixe
  - Opérateurs arithmétiques plus simples
  - Implantation
    - Circuits spécifiques
    - Certains DSPs
  - applications
- Virgule flottante
  - Implantation
    - Processeurs généralistes
    - Certains DSPs
  - Applications
    - Filtrage adaptifs
    - TS (les coefficients ont besoin d'une large dynamique)
    - …


Pads
Input registers
Operand select
C register
Operand select
Timing
Divide/square root registers and multiplexers
Divide/square root state machine
Seed programmable logic array
Pre-alignment
Signed digit multiplier
ALU
Instruction decoder
Pads
Pads
Pipeline register
Pipeline register
Normalizer
Status
Signed digit converter
Rounder
Sum register
Product register
Pads

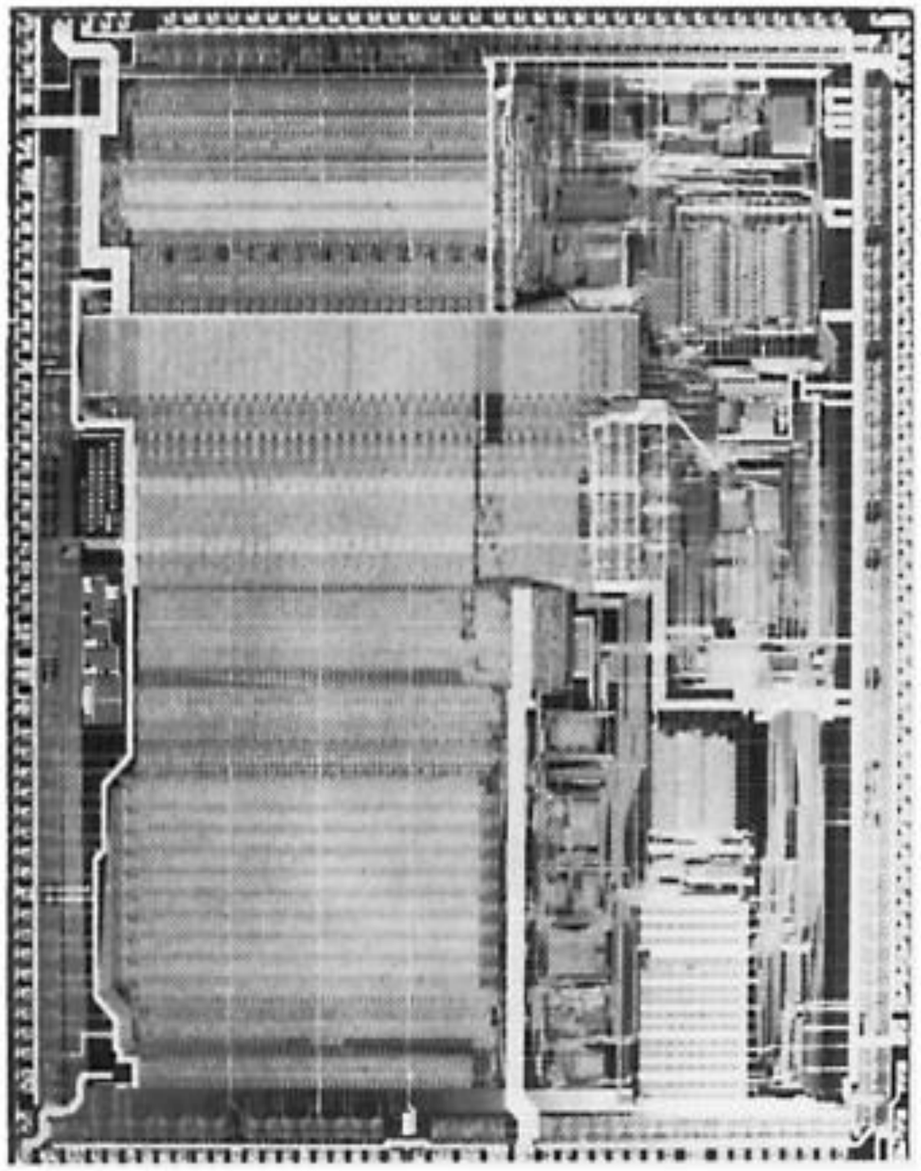
TI 8847

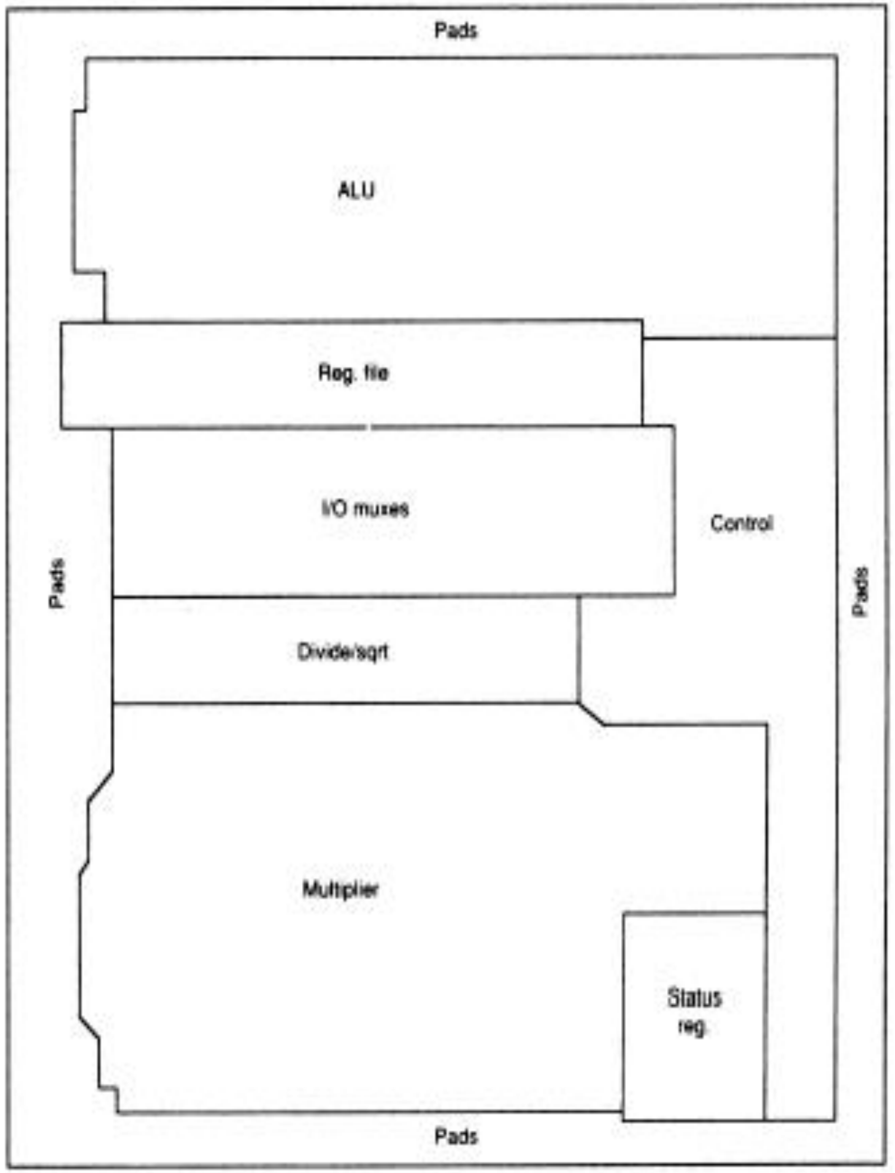

Core Floor Plan
Floating Point Unit
Test/Debug
Data L2 TLB
X86 Decode
Bus Unit
Instruction Cache
L2 Sub Array
Inst TLB/Tag
L2 TAG
Branch Predict
Ucode ROM
ROB
Data Cache
Integer Unit
Data Tag/TLB
Load Store Unit




Pads
Register timing and decode
Register file
IEEE exception and status register
Conflict detect
Forward ing control
Operand fowarding and staging
External data bus
Instruction
decode
32b ↔ 64b alignment
Exponent datapath
Add control
Shifter
Sticky
Pipeline control and interlock logic
Pads
Adder
Pads
IEEE rounding
Divide control
Divider
Sticky
Clocks
Phase locked loop
I/O control
Multiply control
Multiplier
Sticky
Pads

MIPS R3010




Pads
ALU
Reg. file
I/O muxes
Control
Pads
Pads
Divide/sqrt
Multiplier
Status
reg.
Pads

Weitek 3364

Implémentation matérielle

Features	MIPS R3010	Weitek 3364	TI 8847
Clock cycle time (ns)	40	50	30
Size (mil^2)	114,857	147,600	156,180
Transistors	75,000	165,000	180,000
Pins	84	168	207
Power (watts)	3.5	1.5	1.5
Cycles/add	2	2	2
Cycles/mult	5	2	3
Cycles/divide	19	17	11
Cycles/square root	–	30	14

Figure H.36 Summary of the three floating-point chips discussed in this section. The cycle times are for production parts available in June 1989. The cycle counts are for double-precision operations.

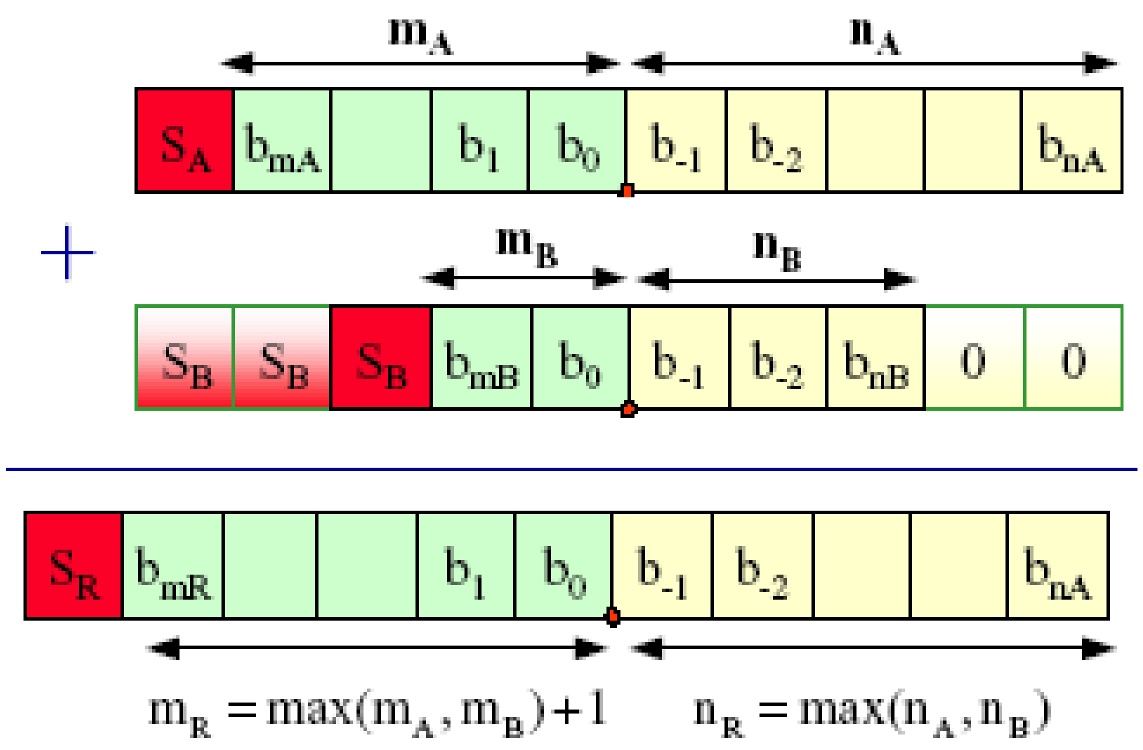
Opérations arithmétiques sur les flottants

- virgule fixe
- virgule flottante

L’addition

- Addition en virgule fixe
- Addition en virgule flottante

Addition virgule fixe



Mécanisme d'addition classique avec extension de signe et ajout de la virgule

Addition en virgule flottante

- Algorithme à suivre
  - Décaler à droite la mantisse du nombre possédant le plus petit exposant jusqu’à arriver à l’exposant de l’autre nombre.
  - Additionner les mantisses
  - Re-Normaliser le résultat
  - Arrondir (éventuellement)

Exemple

- Calcul de 3+1.5 en flottant
- Notation simple selon le format IEEE 754
- a=3, b=1.5

a= 1.10000000000000000000000.2^1
b= 1.10000000000000000000000.2^0

a 0 10000000 10000000000000000000000

Exp= 1+127=128=1000000

b 0 01111111 10000000000000000000000

Exp= 0+127=127=01111111

Alignement des mantisses (on modifie b pour que les 2 nombres aient le même exp)

A=1.1000000000000000000000000.2^1
B=0.1100000000000000000000000.2^1

a	0 10000000 10000000000000000000000
b	0 10000000 11000000000000000000000

Addition des mantisses

S=10.0100000000000000000000000.2^1

Renormalisation

S=1.00100000000000000000000000.2^2

Arrondi

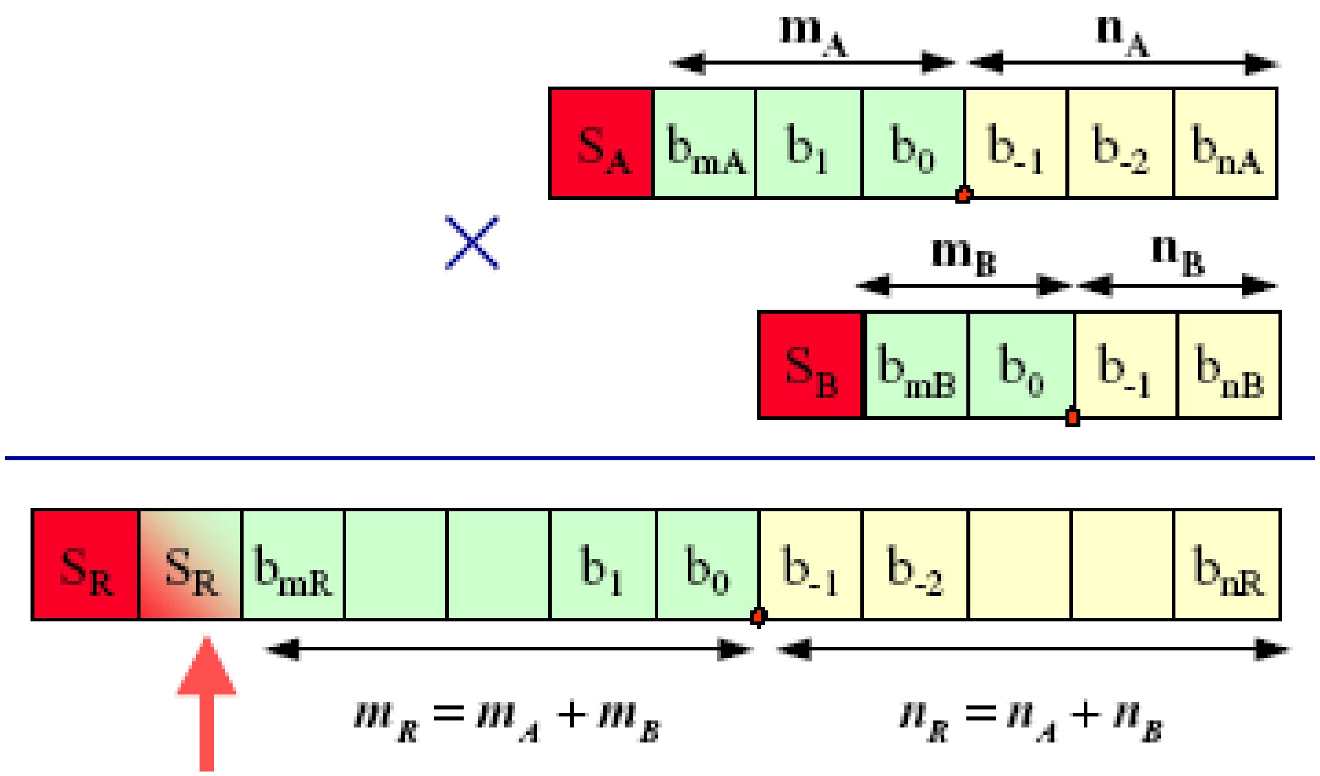
S=1.001000000000000000000000.2^2

s	0 10000001 00100000000000000000000

La soustraction

- Même principe que l'addition puisque
  - A+B = A+(-B)
  - En complément à 2

La multiplication en virgule fixe



Mécanisme de multiplication classique avec ajout de la virgule

La multiplication/division en virgule flottante

- Soit à calculer a*b ou a/b

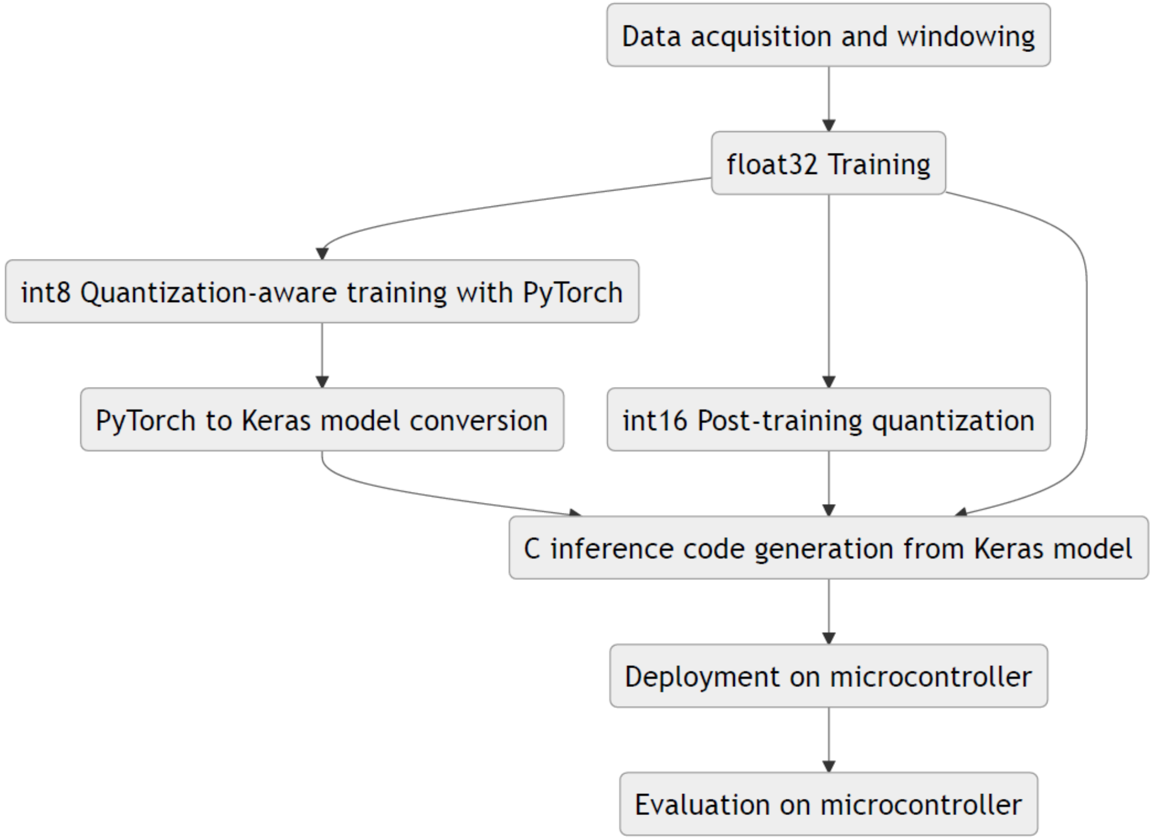
$$p = a \times b = (M_a \times M_b) \times 2^{(ExpA + ExpB)}$$

$$q = \frac{a}{b} = \frac{M_a}{M_b} \times 2^{(ExpA - ExpB)}$$

- Les opérations sur les exposants sont effectuées en complément à 2
- Les opérations sur les mantisses sont effectuées en virgule fixe
- Ne pas oublier les éventuels dépassements de capacité

IA on the Edge : Deployment on MCU

MicroAI General Flow



Pierre-Emmanuel Novac, Ghouthi Boukli Hacene, Alain Pegatoquet, Benoît Miramond, Vincent Gripon. Quantization and Deployment of Deep Neural Networks on Microcontrollers. *Sensors*, MDPI, 2021

Dark side UCA Board
RFT dev board

- The target is a **RFT-AI Dev. Kit** board equipped with a **STM32L476RGT6 Microcontroller.** This MCU is based on **the ARM Cortex M4 architecture** and runs at a frequency of **80 MHz. The board provides 1 MB Flash and 128 KB SRAM.**
  - LoRa SX1262 Module with CP antenna
  - Quectel L96 M33 GPS module
  - Accelerometer
  - Gyroscope
  - Magnetometer
  - 9 Axis Sensor TDK InvenSense ICM-20948 - Digita
  - PDM Microphone MEMS (Silicon) Omnidirection SPH0690LM4H
  - Air Quality Sensor - Sensirion AG SGP30-2.5K
  - Optical Sensor Ambient - Lite-On Inc. LTR-303ALS