SEI - SoC - CNN

S. Mancini

Grenoble INP phelma

Plan

- Projet d'intégration des SoC
- Les CNN
- Détails d'organisation
- CNN cible : CIFAR10

Objectifs

Réaliser un système de traitement HW et SW, et arriver jusqu'à une implémentation avec caméra et affichage

Mettre en oeuvre toutes les méthodes et outils dont vous avez besoin.

Votre rôle

Vous avez tous le même algorithme, et vous partez d'une page blanche.

Votre mission est:

- Spécifier la solution
- Définir les techniques de résolution
- Mettre en oeuvre la solution
- Evaluer vos résultats, selon les critères usuels

⇨ Votre attitude est déterminante et vous devez chercher des solutions par vous même.

Notre rôle

- Nous vous donnons des indications méthodologiques
- Nous vous dépannons sur les outils ...
- ... lorsque vous avez exploré par vous même

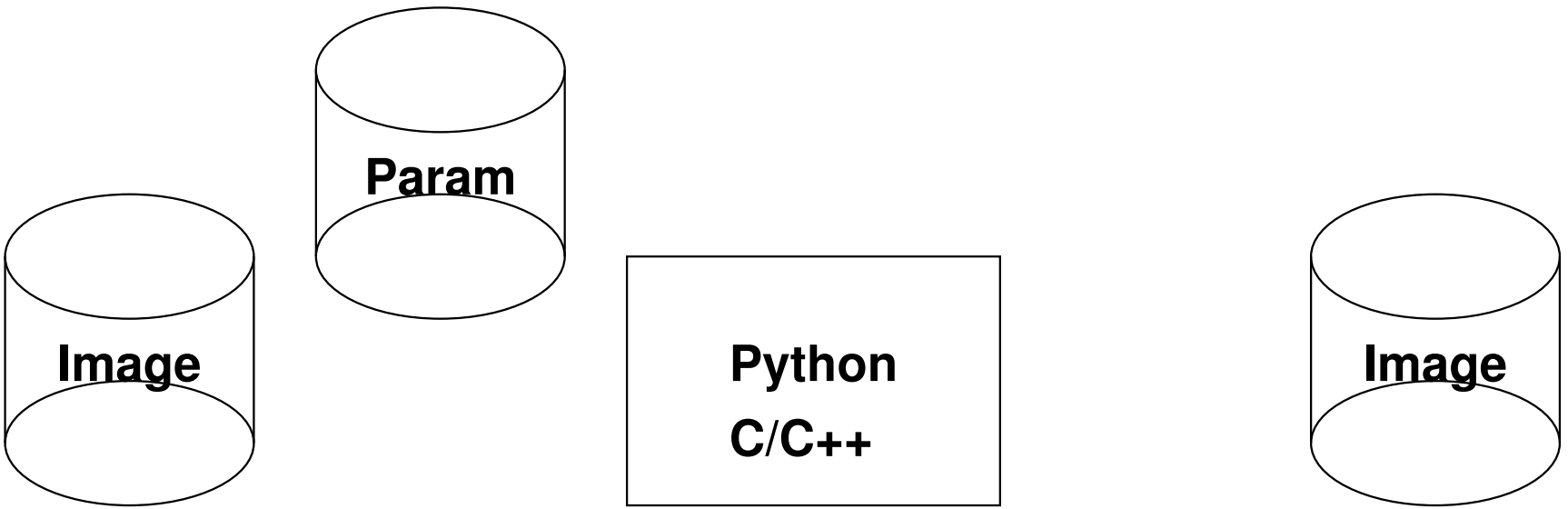
Les connaissances dont vous avez besoin

... et que vous avez !

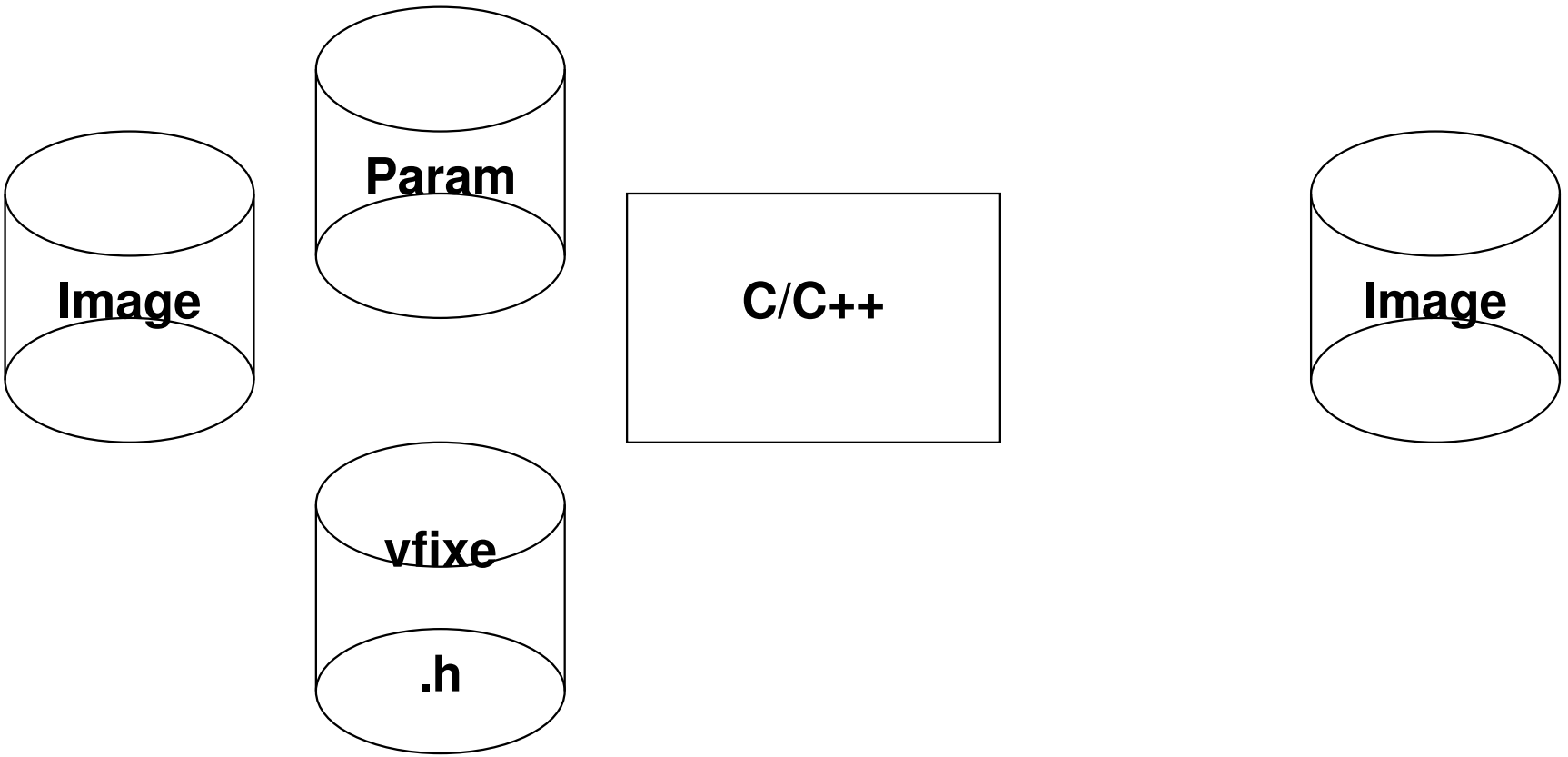
- Passer d'une spécification algorithmique à une architecture de traitement HW ou SW
- Modélisation RTL (VHDL) et HLS (CatapultC)
- Informatique embarqué (C)
- Conception FPGA Xilinx ou Altera
- Outils informatiques usuels (Linux)
- Python vous sera d'une grande aide

Algorithme !

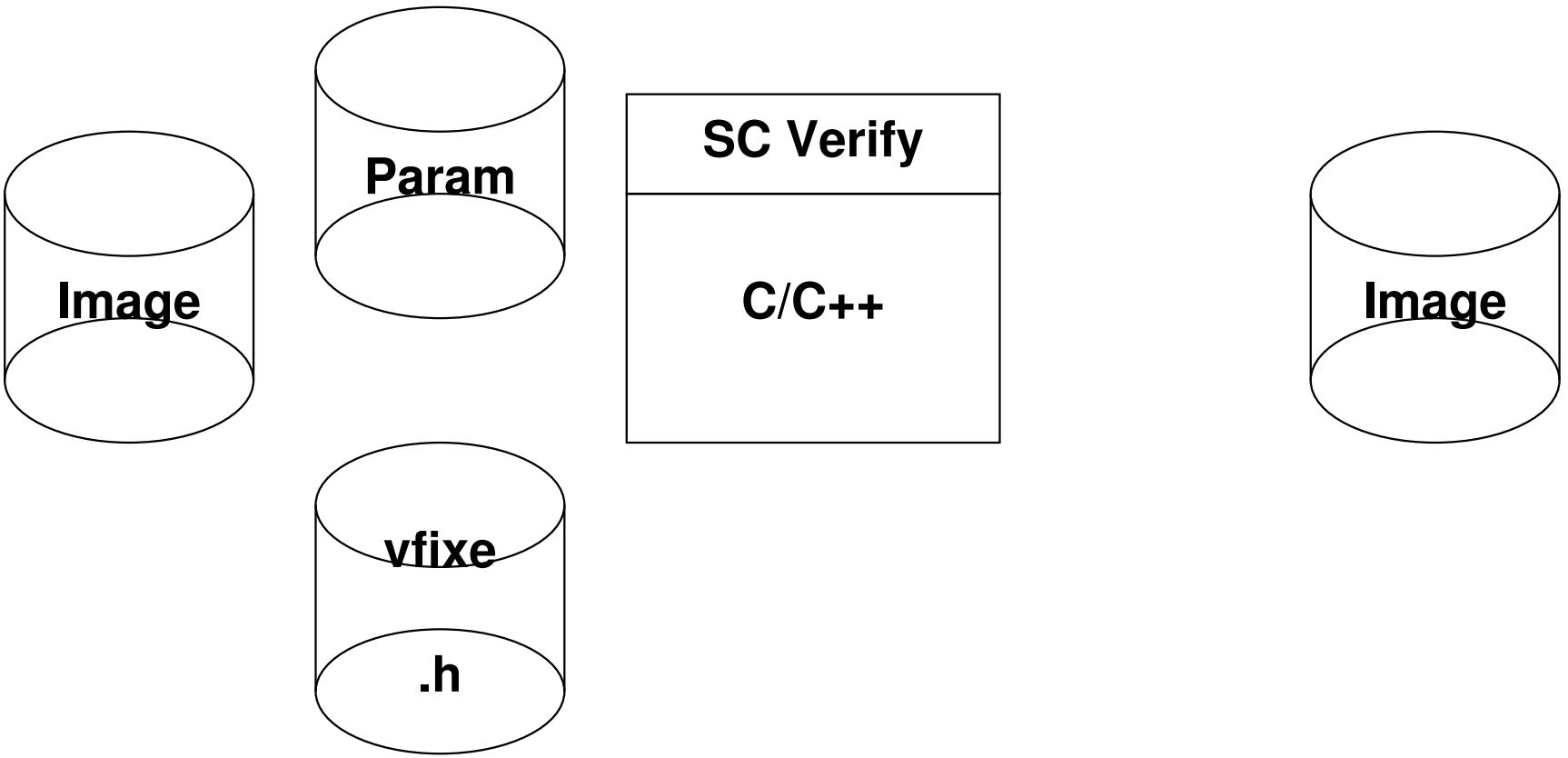
Etapes de la méthodologie HW



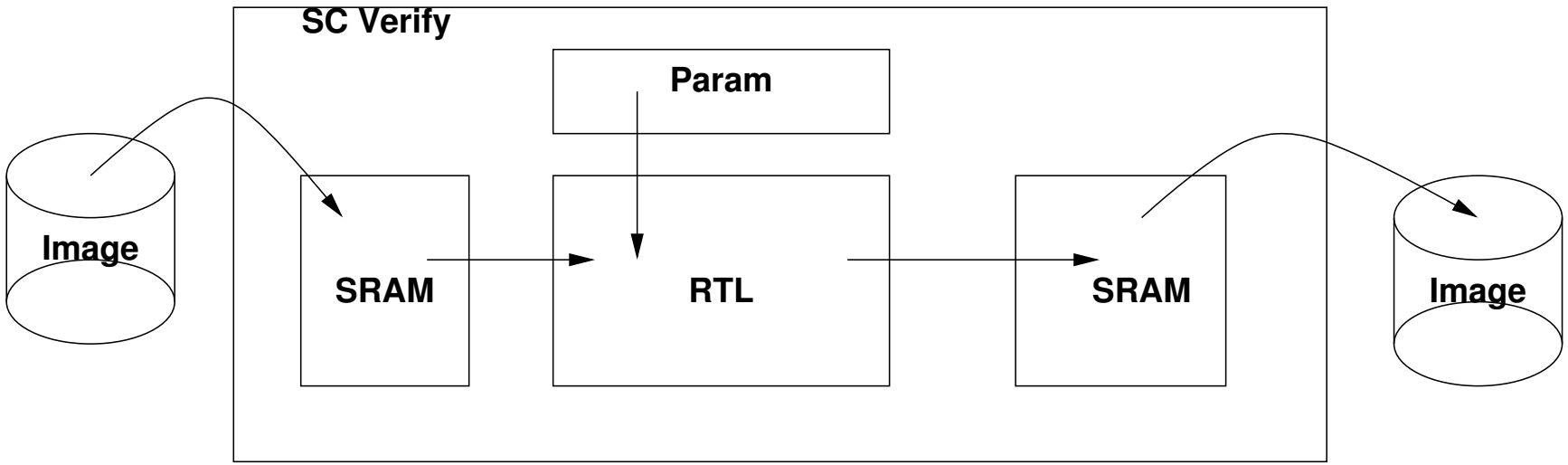
Etapes de la méthodologie HW



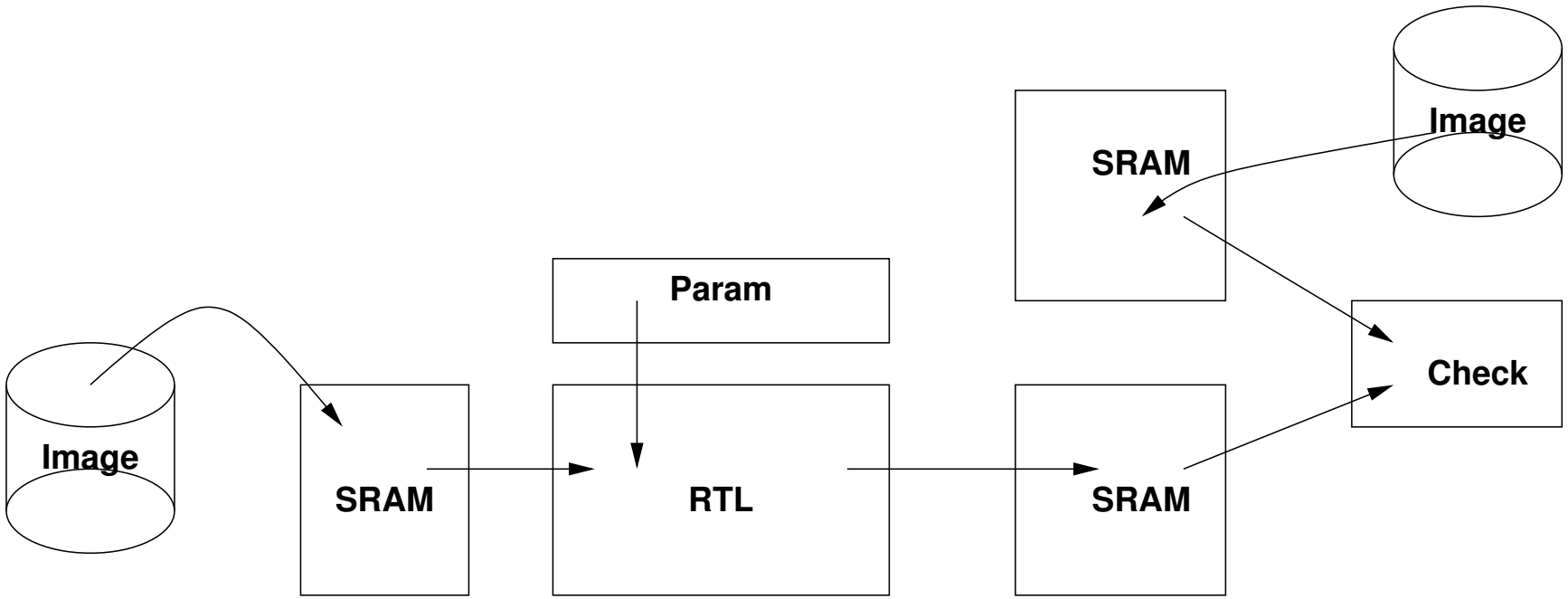
Etapes de la méthodologie HW



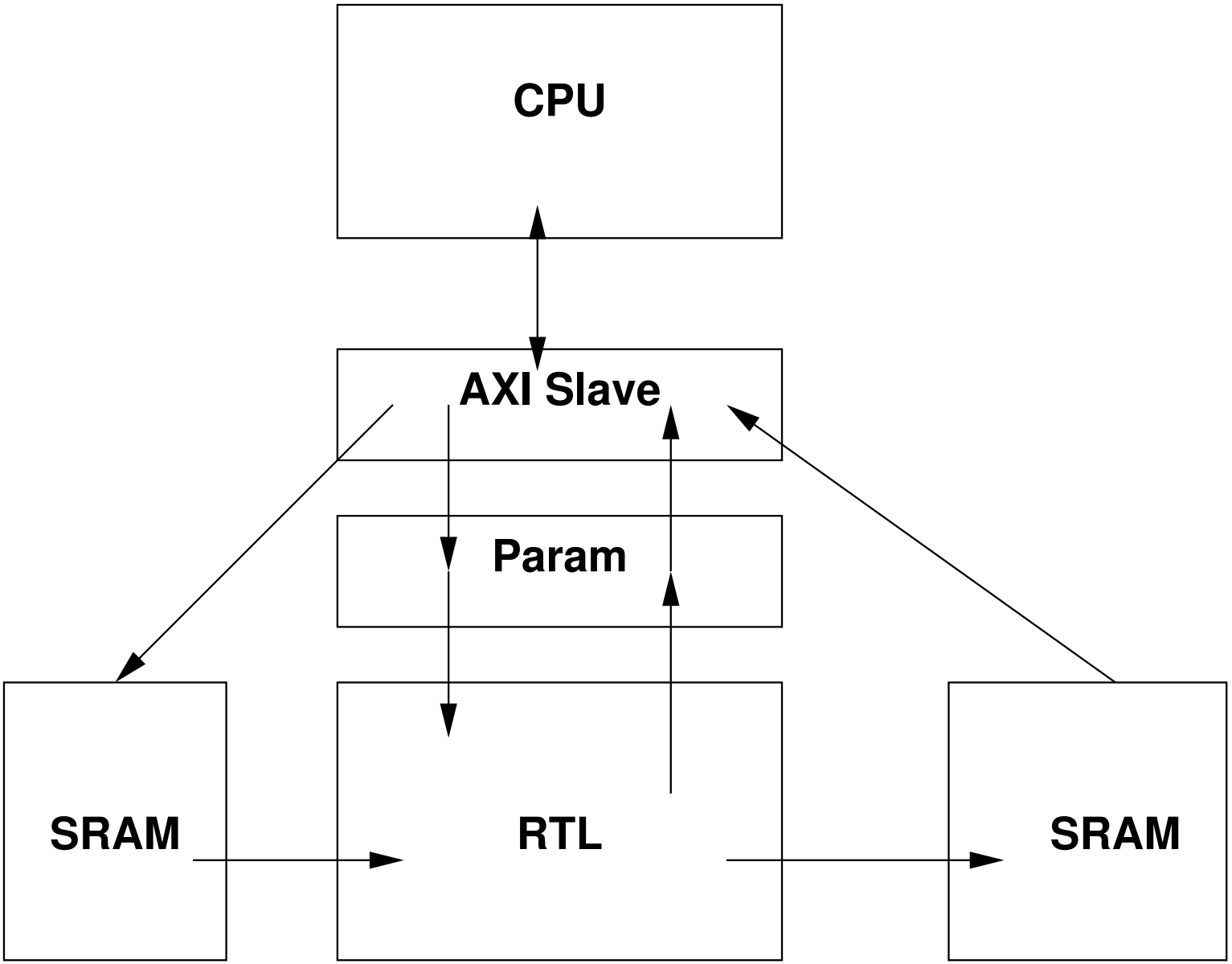
Etapes de la méthodologie HW



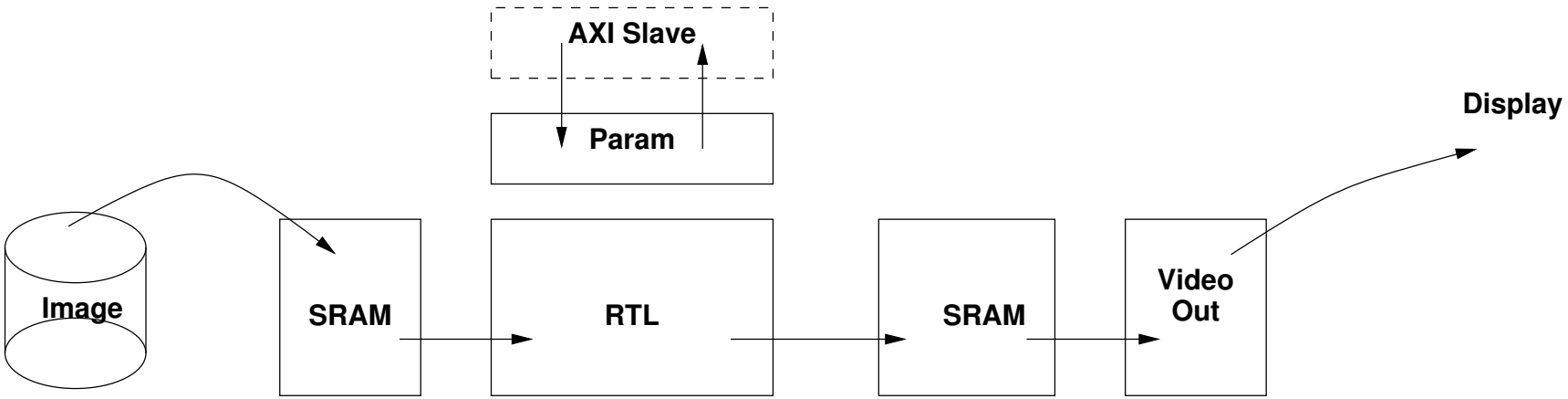
Etapes de la méthodologie HW



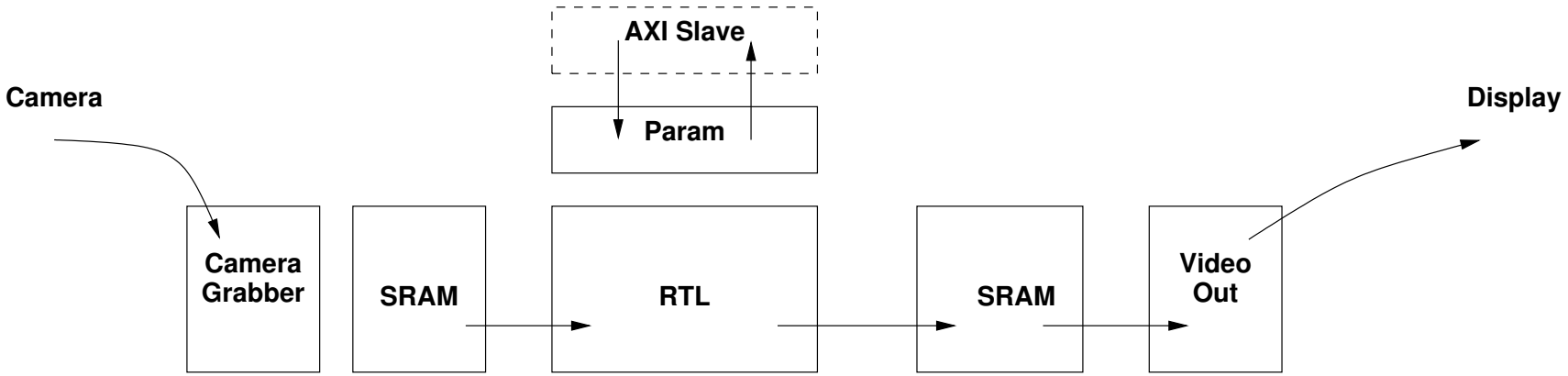
Etapes de la méthodologie HW



Etapes de la méthodologie HW



Etapes de la méthodologie HW



Plan

- Projet d'intégration des SoC
- Les CNN
- Détails d'organisation
- CNN cible : CIFAR10

Fonctions génériques 1/2

Une tâche d'IA consiste à déterminer les données 'génératrices' x qui ont produit une 'observation' y (mesure). Par exemple, la donnée génératrice 'chat' génère toutes les images possibles de chats !

Hypothèse : il existe une fonction f qui permet de passer d'une 'observation' y à ses données 'génératrices' x:

$$x = f(y)$$

Problème: f est inconnue à-priori.

Le "deep-learning" consiste à chercher f à partir d'un ensemble d'observations, qui est la base de données d'apprentissage: on connaît des données $\mathbf{y}$ et x associées.

Fonctions génériques 2/2

On construit f à partir de fonctions génériques paramétrables de façon systématique:

$$f(\theta, y) = f_{n-1}(\theta_{n-1}, f_{n-2}(\theta_{n-2}, \dots f_0(\theta_0, y)))$$

Les fonctions f_n sont les “couches” du réseau:

- Fonctions linéaires
- Couches ’totalement’ connectées : le perceptron
- Fonctions non-linéaires dites **d’activation** (sigmoïd, RELU, maxpool, softmax, etc . . .)
- Convolution
- Sous-échantillonnage

L’apprentissage consiste à déteminer $\theta = (\theta_0, \dots, \theta_{n-1})$ pour un jeu de données $(x, \mathbf{y})$

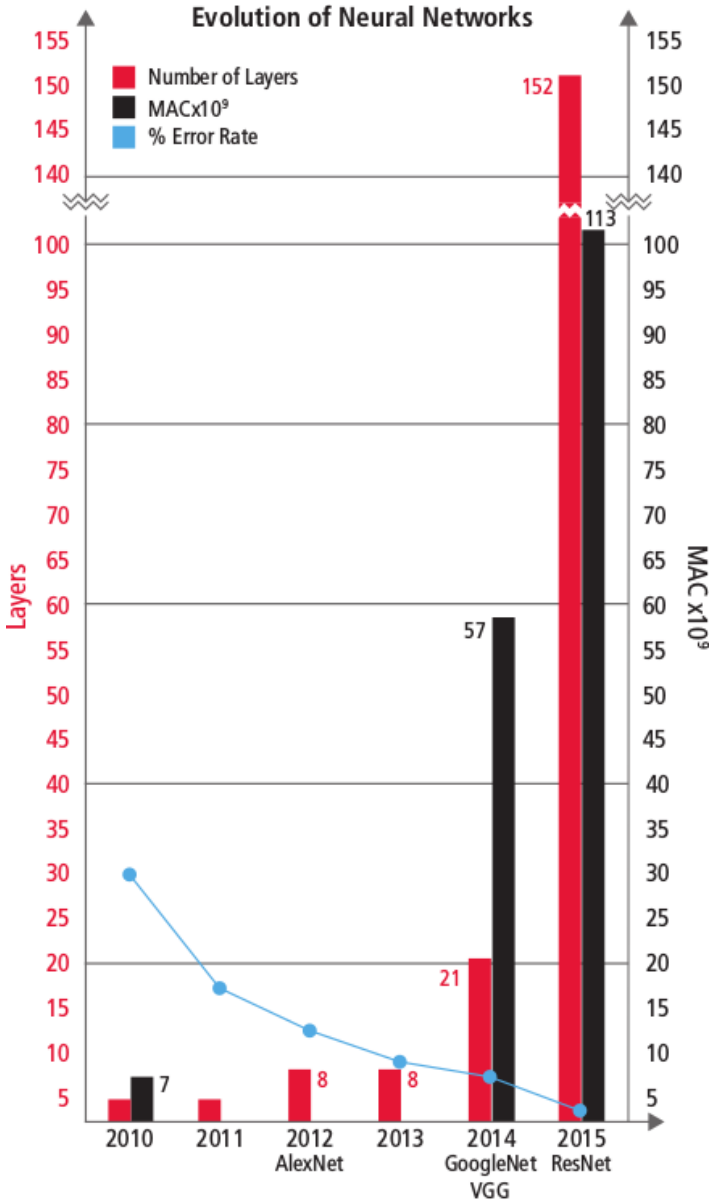
Un réseau de neurones ?

- Un objectif : la mission du réseau de neurones
- Une base de données d'apprentissage
  - Pour la configuration
  - et le test
- Une méthode d'apprentissage
  - Fonction de coût
  - Régularisation
  - Méthode d'optimisation
- Une 'architecture' de réseau
  - Nombre et nature des couches

CNN

Les CNN, ou Convolutional Neuron Network, sont une technique de Deep Learning pour de nombreuses tâches de **traitement d'image**: classification, détection, identification, amélioration de qualité, flot optique...

Un CNN est constitué seulement des couches de convolution, sous-échantillonnage, "pooling" et RELU.



Convolution

Un **kernel** K de **convolution** est appliqué sur une image:

$$S(i,j) = (K \times I)(i,j) = \sum_m \sum_n I_{i+m,j+n} K_{m,n}$$

En réalité, à un étage, une image est 'multi-canal' et chaque pixel a plusieurs 'composantes':

$$S(c,i,j) = \sum_{l,m,n} I_{l,i+m,j+n} K_{c,l,m,n}$$

Un canal de sortie c est produit à partir de tous les canaux l de l'image d'entrée

RELU

RELU pour *REctified Linear Unit* est la fonction d'activation non-linéaire des CNN.

$$RELU(a) = \max(0, a)$$

Par rapport à la sigmoïd du perceptron, les avantages de RELU sont:

- Facile à implémenter
- Dérivable (presque) partout, à dérivée non-nulle même pour les valeurs extrêmes
  - ⇨ L'apprentissage est facilité

MaxPool

Maxpool est l'autre fonction non-linéaire de choix:

- Sous-échantillonnage pour la réduction des données
- Remplace un groupe de données par son maximum (ou médian, etc . . .)

SoftMax

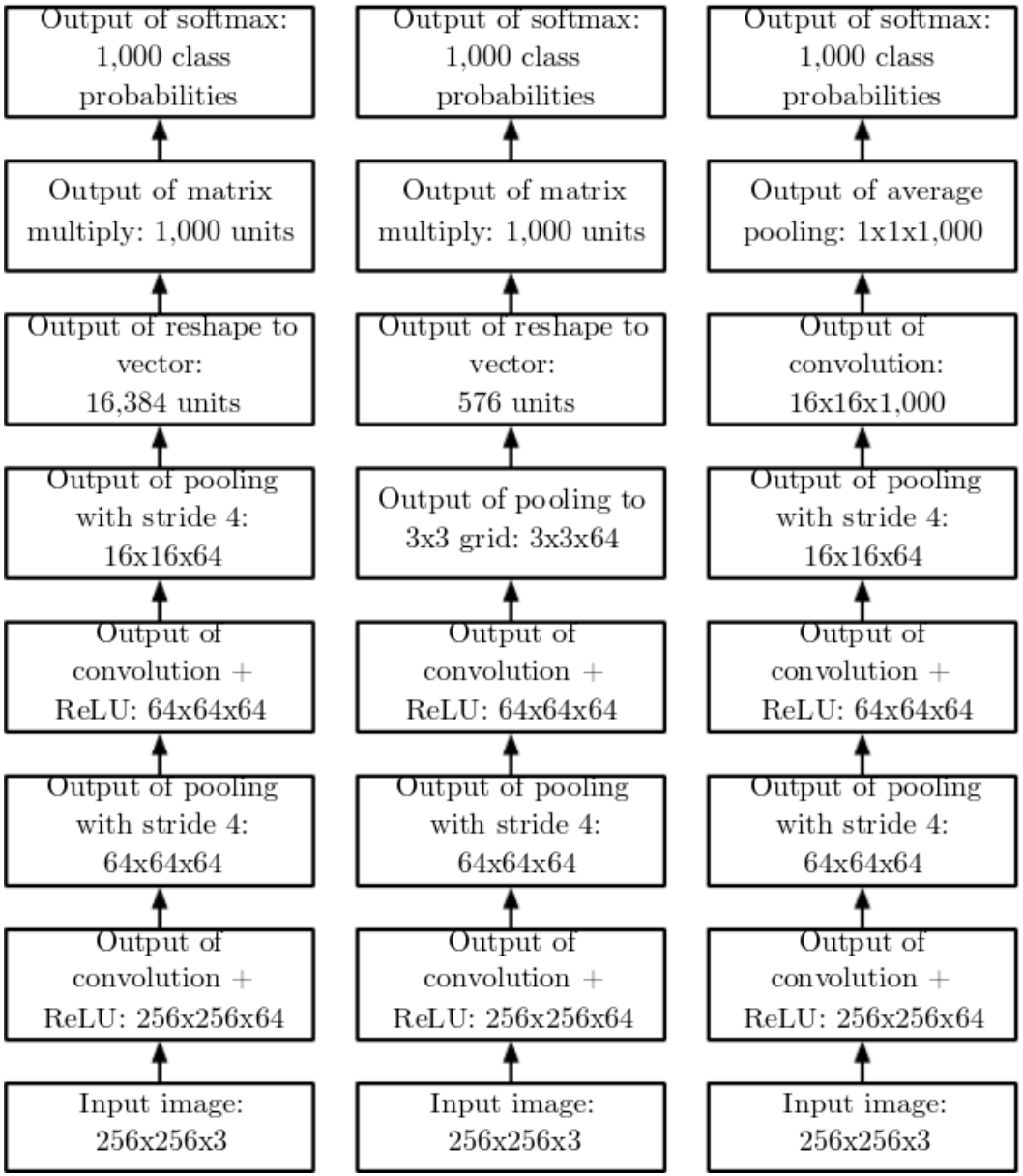
Softmax est utilisé pour les tâches de classification car le CNN ne produit pas de valeurs binaires telles qu'une appartenance à une classe

$$\mathrm{softmax}(z)_i = \frac{\exp z_i}{\sum_j \exp z_j}$$

Avec z un vecteur de probabilité pour chacune des classes potentielles. $\mathrm{softmax}(z)_i$ fait 'émerger' la classe la plus probable.

Remarque: **Softmax** provient du monde bayesien et est lié aux techniques d'optimisations statistiques basées sur le maximum de log-vraisemblance (log-likelyhood)

Exemples de CNN



Plan

- Projet d'intégration des SoC
- Les CNN
- Détails d'organisation
- CNN cible : CIFAR10

Documentation et fichiers dans

`/tp-fmr/smancini/SEI_SoC_CNN/`

Méthode

- Valider votre logiciel ou matériel sur des jeux de coefficients ’fictifs’, dont les valeurs vous permettent de vérifier les calcul
- Utiliser des coefficients appris

La deuxième phase démarre mi-Décembre

Organisation

Projet par **binôme**

- Séances : 40 heures
- Au moins 24 heures non encadrées

Le vendredi matin, jusqu'à fin Janvier.

Evaluation:

- Présentation intermédiaire : 10 mn, le **6/12**
- Rapport + présentation finale : 10 mn , à la dernière séance
- Réalisation minimale pour valider le module: convolution d'image sur FPGA, flux vidéo 'live'

Plan

- Projet d'intégration des SoC
- Les CNN
- Détails d'organisation
- CNN cible : CIFAR10

CIFAR 10 database

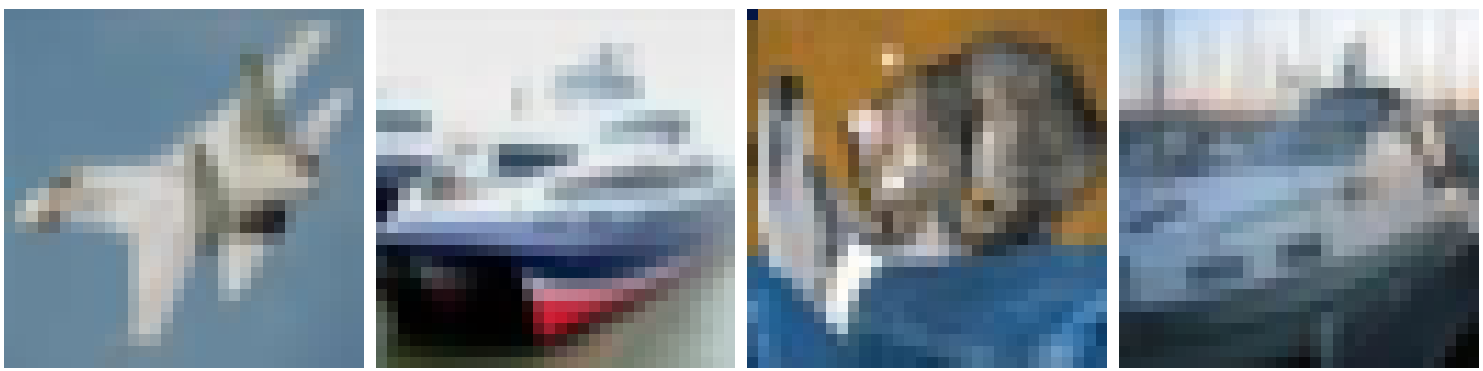
Binary version

The binary version contains the files `data_batch_1.bin, data_batch_2.bin, ..., data_batch_5.bin,` as well as `test_batch.bin`. Each of these files is formatted as follows:

```
<1 x label><3072 x pixel>
...
<1 x label><3072 x pixel>
```

In other words, the first byte is the label of the first image, which is a number in the range 0-9. The next 3072 bytes are the values of the pixels of the image. The first 1024 bytes are the red channel values, the next 1024 the green, and the final 1024 the blue. The values are stored in row-major order, so the first 32 bytes are the red channel values of the first row of the image.

Each file contains 10000 such 3073-byte "rows" of images, although there is nothing delimiting the rows. Therefore each file should be exactly 30730000 bytes long.



CIFAR 10 : utilitaires

Fichiers dans

`/tp-fmr/smancini/SEI_SoC_CNN/cifar10_data`

Exemple de script pour lire une image:

```
nb=n*3073+1
cat cifar10_data/cifar-10-batches-bin/test_batch.bin |
        dd count=3072 bs=1 skip=<n> > toto.raw
display -size 32x32 -depth 8 -interlace Plane  rgb:toto.raw
convert -size 32x32 -depth 8 -interlace Plane  rgb:toto.raw cifar10_voilier.png
convert -size 32x32 -depth 8 -interlace Plane  rgb:toto.raw cifar10_voilier_bin.ppm
convert -size 32x32 -depth 8 -interlace Plane  rgb:toto.raw -compress none cifar10_voilier
```

Adaptation des images

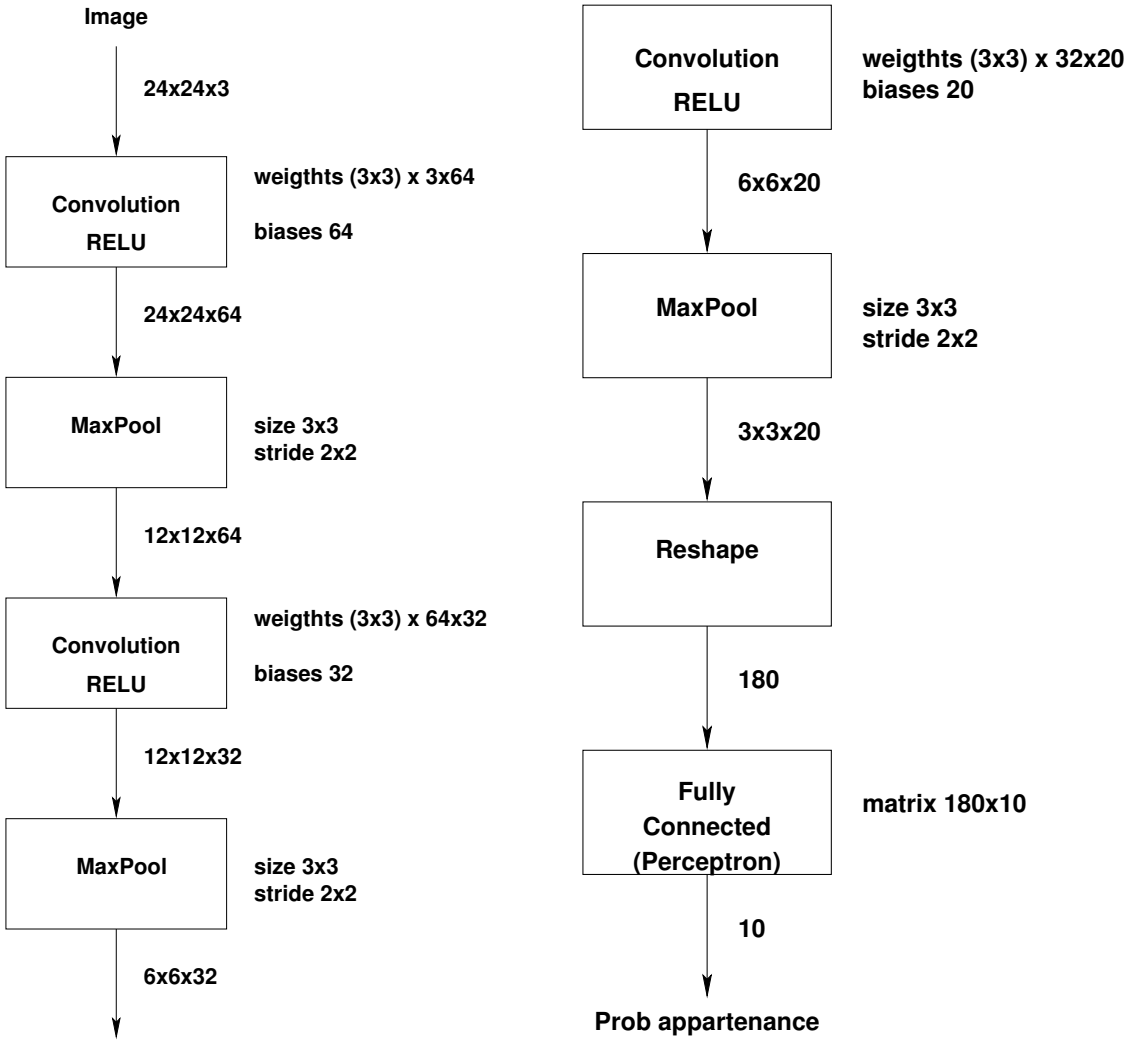
Les images 32x32 sont 'coupées' à 24x24, la coupe étant centrée.

Attention: pour la détection, une image m doit être normalisée:

$$\mu = \frac{1}{N}\sum m_{i,j} \quad ; \quad \sigma = \sqrt{\frac{1}{N}\sum (m_{i,j} - \mu)^2}$$

$$m'_{i,j} = (m_{i,j} - \mu) / \max(\sigma, \frac{1}{\sqrt{N}})$$

CNN 'basique' pour CIFAR10



reshape : transforme la matrice en vecteur, dans l'ordre cannonique
Format des données : **HWC** HeigthxWidthxCanal
Autrement dit, les 'canaux' sont en 'premier'.

CNN 'basique' pour CIFAR10

Les coefficients sont dans
`CNN_coeff_3x3.txt` et
`CNN_coeff_5x5.txt`

CNN 'basique' pour CIFAR10

TODO List

- Python : code de base pour lecture/ecriture fichier pgm, ecriture coe
- Catapult : Options de base (techno, IO)
  - Altera : biliothèque de mémoires simple & double port
- Modelsim : config qui marche (ok, voir script)
- Vivado :

  interfaces mémoire et affichage

  reset de l'IP CC

  Lien Vivado CC pour paramétrage memoires

  RGB par mot ou par plan
- projet SW : affichage VGA
- CNN :

  détail structure des données intermédiaires (dernier perceptron)