Programmation CUDA : la convolution

1 Introduction

La convolution est une opération de base en traitement du signal et en intelligence artificielle connexionniste. Les premières couches d'un réseau de neurones profond réalisent souvent cette opération afin d'effectuer un filtrage des données. Si l'on se donne un vecteur d'entrée $X = (x_0, \ldots, x_{n-1})$ et un vecteur de coefficients $H = (h_0, \ldots, h_{p-1}), p \leq n$, le vecteur convolué $Y = X \star H$ se définit comme suit :

$$y_j = \sum_{k=\max(0,j-p)}^{j} x_k h_{j-k},\ j = 0 \ldots n-1 \tag{1}$$

Dans le cas bidimensionnel, X et H sont des matrices de dimensions respectives $m \times n$ et $p \times q$ et l'équation de convolution $Y = X \star H$ devient :

$$y_{i,j} = \sum_{k=\max(0,i-p)}^{i} \sum_{l=\max(0,j-q)}^{j} x_{i,j} h_{i-k,j-l},\ i = O \ldots m-1,\ j = 0 \ldots n-1 \tag{2}$$

2 Approche naïve

2.1 Noyau de calcul

En s'inspirant de ce qui a été vu en cours pour le produit matriciel, proposer un noyau CUDA permettant de calculer le produit de convolution de deux vecteurs et de deux matrices. Les signatures des deux fonctions seront :

```
__global__void conv1D(int n, float *x, int p, float *h, float *y);
__global__ void conv2D(int m, int n, float *x, int p, int q, float *h, float *y);
```

2.2 Appel du noyau

Tester le bon fonctionnement du noyau en générant des vecteurs et des matrices aléatoires et en comparant avec les résultats obtenus sur CPU. Déterminer approximativement les performances en GFLOPS du produit de convolution sur CPU et GPU.

3 Codage par blocs

On peut améliorer les performances avec la mémoire partagée. Proposez un nouveau noyau exploitant au mieux la localité des données et comparez avec l'approche naïve.