Évaluation homomorphe de réseaux de neurones convolutionnels

CKKS, chiffrement homomorphe, CNN

by

Siméon Laporte
Stagiaire
ENS Rennes
Bruz

Guillaume Hanrot
Maître de stage
FHE Lab
Lyon

Abstract :
L'efficacité des algorithmes homomorphes est importante pour pouvoir les utiliser en pratique. On propose ici deux techniques accélérant l'évaluation homomorphe des couches convolutionnelles de réseaux de neurones (CNN), elles se basent toutes les deux sur une avancée récente [1] qui accélère grandement le produit homomorphe d'une matrice claire avec une matrice chiffrée (PC-MM) dans le schéma de chiffrement CKKS. Dans ces deux techniques, on réduit l'étape de convolution d'un CNN à un produit matriciel. On tire aussi parti des propriétés SIMD (Single Instruction Mutiple Data) du schéma CKKS pour évaluer la convolution de plusieurs images en même temps.

§1. Introduction

Le chiffrement homomorphe (HE) est un ensemble de techniques rendant possibles les calculs sur des données chiffrées sans les déchiffrer [2]. En particulier, les systèmes de chiffrement homomorphes peuvent être une solution pour préserver la vie privée de l'utilisateur de services délocalisant des calculs, comme par exemple des systèmes de MLaaS (Machine Learning as a Service) rendant les services d'intelligence artificielle accessibles à distance au grand public.

Dans ce rapport, on se concentrera sur l'accélération de l'évaluation homomorphe d'un certain type de systèmes d'intelligence artificielle : les réseaux de neurones convolutionnels (CNN), et particulièrement de leur étape de convolution. Le schéma de chiffrement homomorphe que l'on utilise est CKKS [3], que l'on présente dans la suite.

Les CNN sont particulièrement utilisés dans l'analyse et la classifications d'images, ils sont composés de plusieurs couches : les couches convolutionnelles qui sont des couches de combinaisons linéaires locales, et de couches d'activation.

On se concentrera dans la suite de ce rapport sur l'accélération des couches convolutionnelles. Et nous prendrons pour nos expériences les valeurs des couches de convolution d'un CNN bien connu : ResNet18 [4].

§2. Chiffrement homomorphe

Un schéma de HE a la même forme qu'un schéma de chiffrement cryptographique classique, mais dispose aussi d'une fonction Eval et d'une clef d'évaluation qui permet d'évaluer de manière homomorphe des cricuits arithmétiques. [2]

Définition 2.1 (HE) : Un schéma de chiffrement homomorphe (HE) est donné par 4 ensembles ($\mathcal{M}, \mathcal{P}, \mathcal{C}$ et $\mathcal{K}$, respectivement les ensembles des messages clairs, des plaintexts, des chiffrés et des clefs) et par plusieurs algorithmes :

- $\text{Ecd} : \mathcal{M} \to \mathcal{P}$ (fonction d'encodage)
- $\text{Dcd} : \mathcal{P} \to \mathcal{M}$ (fonction de décodage)
- $\text{Enc} : \mathcal{K} \times \mathcal{P} \to \mathcal{C}$ (fonction de chiffrement)
- $\text{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{P}$ (fonction de déchiffrement)
- $\text{Eval} : \mathcal{K} \times \mathcal{C} \times \{\text{circuits}\} \to \mathcal{C}$ (fonction d'évaluation homomorphe)

Eval est telle que si $\text{Eval}(s, \text{ct}, C) = \text{ct}'$, alors $C(\text{Dec}(s, \text{ct})) = \text{Dec}(s, \text{ct}')$.
On a aussi que $\text{Dcd} \circ \text{Ecd} = \text{Id}$ et que $\text{Dec}(s, \text{Enc}(s, m)) = m$

Note 2.1: La vision d'un schéma de HE avec une fonction Eval est très théorique. En pratique, un schéma de HE fournit un nombre limité d'opérations homomorphes, qu'on peut ensuite combiner pour obtenir des circuits plus complexes.

§3. Le schéma CKKS

Notation 3.1: Dans tout ce rapport, $N \in \mathbb{N}$ est une puissance de 2. On notera log le logarithme en base 2.
On pose $\mathcal{R}_N = \frac{\mathbb{Z}[X]}{(X^N+1)}$ et pour $Q \in \mathbb{N}^*$, on pose $\mathcal{R}_{Q,N} = \frac{\mathcal{R}_N}{Q\mathcal{R}_N}$.

§3.1. RLWE

Définition 3.1.1 (Problème RLWE (version décisionnelle)):
Étant donné une liste de couples $(a_i, b_i) \in \mathcal{R}_{Q,N}$, décider si elle est issue de la loi uniforme sur $\mathcal{R}_{Q,N}^2$ ou si elle suit la loi de $(a, -as_k + e)$ avec a uniforme sur $\mathcal{R}_{Q,N}$, e et s_k sont aléatoires et petits.

Le problème RLWE fonde la sécurité de plusieurs systèmes modernes de chiffrement. Le schéma CKKS en particulier, se base sur ce problème [3].

§3.2. CKKS [3]

Soient les ensembles $\mathcal{M} = \mathbb{C}^{\frac{N}{2}}, \mathcal{P} = \mathcal{R}_{Q,N}, \mathcal{C} = \left(\mathcal{R}_{Q,N}\right)^2$ et $\mathcal{K} = \mathcal{R}_N$

CKKS manipule des vecteurs de réels de taille N, qu'on considérera généralement comme des vecteurs de nombres complexes de taille $\frac{N}{2}$ (d'où $\mathcal{M} = \mathbb{C}^{\frac{N}{2}}$).

Note 3.2.1: Posons pour la suite de ce rapport $\left(\zeta_j\right)_{j\in[\![0,\frac{N}{2}-1]\!]} = \left(e^{2i\pi 5^j/2N}\right)_{j\in[\![0,\frac{N}{2}-1]\!]}$ une famille de $\frac{N}{2}$ racines $2N$-ème de l'unité non conjuguées deux à deux.
La fonction :

$$\begin{aligned}\mathcal{R}_N &\to \mathbb{C}^{\frac{N}{2}}\\ P &\mapsto \left(P(\zeta_i)\right)_{i\in[\![0,\frac{N}{2}-1]\!]}\end{aligned}$$

est un morphisme injectif entre $\mathcal{R}_N$ et $\mathbb{C}^{\frac{N}{2}}$, qu'on notera DFT en référence à la Transformée de Fourier Discrète. On notera sa réciproque iDFT.

Définition 3.2.1 (fonctions d'encodage): Avant de chiffrer, on a dans CKKS une première étape d'encodage, on a alors les fonctions d'encodage et de décodage suivantes :

- $\begin{aligned}\mathrm{Ecd}_{\mathrm{slot}} : \mathbb{C}^{\frac{N}{2}} &\to \mathcal{P}\\ (z_i)_{i\in[\![0,\frac{N}{2}-1]\!]} &\mapsto \lfloor \Delta\ \mathrm{iDFT}(z)\rceil\end{aligned}$
- $\begin{aligned}\mathrm{Dcd}_{\mathrm{slot}} : \mathcal{P} &\to \mathbb{C}^{\frac{N}{2}}\\ P &\mapsto \tfrac{1}{\Delta}\ \mathrm{DFT}(P)\end{aligned}$

Note 3.2.2: On retrouve bien que $\mathrm{Dcd}_{\mathrm{slot}} \circ \mathrm{Ecd}_{\mathrm{slot}} \simeq \mathrm{id}_{\mathbb{C}^{\frac{N}{2}}}$. En effet, le système CKKS est un système approché.

Définition 3.2.2 (Chiffrement): Dans le schéma CKKS, les fonctions de chiffrement et de déchiffrement sont :

- $\begin{aligned}\mathrm{Enc} : \mathcal{K}\times\mathcal{P} &\to \mathcal{C}\\ (s_k, m) &\mapsto (a, -as_k + m + e)\end{aligned}$ où $\begin{cases} a\in\mathcal{R}_Q \text{ est choisi uniformément}\\ e\in\mathcal{R}_Q \text{ est une petite erreur aléatoire}\end{cases}$
- $\begin{aligned}\mathrm{Dec} : \mathcal{K}\times\mathcal{C} &\to \mathcal{P}\\ (s_k, (a, b)) &\mapsto b + as_k\end{aligned}$

On retrouve alors bien $\mathrm{Dec}(s_k, \mathrm{Enc}(s_k, m)) = m + e \simeq m$.
Le schéma est donc approximativement correct.

Note 3.2.3 (Sécurité): La difficulté du problème RLWE et donc la sécurité du schéma CKKS nécessite un couple de paramètres (Q, N) avec Q pas trop grand à N fixé [5].

Théorème 3.2.1 (Homomorphie du schéma): Le schéma CKKS fournit les opérations homomorphes suivantes :

- Add : Prenant en entrée deux chiffrés c_1 et c_2 sous la même clef s_k, Add renvoie un chiffré c_3 sous la clef s_k tel que $\text{Dec}(s_k, c_3) \simeq \text{Dec}(s_k, c_1) + \text{Dec}(s_k, c_2)$
- ScalMult : Prenant en entrée un chiffré c_1 et un scalaire λ, ScalMult renvoie un chiffré c_2 tel que $\text{Dec}(s_k, c_2) \simeq \lambda\ \text{Dec}(s_k, c_1)$
- PMult : Prenant en entrée un chiffré c_1 et un plaintext m, PMult renvoie un chiffré c_2 tel que $\text{Dec}(s_k, c_2) \simeq m \times \text{Dec}(s_k, c_1)$
- CMult : Prenant en entrée deux chiffrés c_1 et c_2 sous la même clef s_k, CMult renvoie un chiffré c_3 tel que $\text{Dec}(s_k, c_3) \simeq \text{Dec}(s_k, c_1) \times \text{Dec}(s_k, c_2)$
- Rot : Prenant en entrée c_1 un chiffré sous la clef s_k tel que $\text{Dec}(s_k, c_1) = \text{Ecd}_{\text{slot}}\left(x_0, ..., x_{\frac{N}{2}-1}\right)$, et un entier i, Rot renvoie un chiffré c_2 sous la clef s_k tel que $\text{Dec}(s_k, c_2) = \text{Ecd}_{\text{slot}}\left(x_i, ..., x_{\frac{N}{2}+1}, x_0, ..., x_{i-1}\right)$.

Note 3.2.4 (Niveaux et Bootstrap): Puisque l'encodage nécessite une étape de *scaling* (par un facteur Δ), la multiplication de deux encodés/chiffrés $(u, v) \mapsto u \cdot v$ doit être corrigé en $(u, v) \mapsto \frac{u \cdot v}{\Delta}$, ce qui a pour effet de diviser le module Q par Δ. On dit qu'on perd un niveau. Dans la mesure où le module ne peut pas être baissé indéfiniment, on a un nombre de niveaux limité pour effectuer les calculs.
Une fois tous les niveaux épuisés, on peut restaurer le module (et ainsi récupérer tous les niveaux) grâce à l'opération dite de Bootstrapping, qui est très coûteuse en temps.
On essaiera alors dans la suite d'utiliser le moins de niveaux possible, de sorte à réduire autant que possible le nombre de Bootstrapping.

Ces opérations homomorphes ont chacune un coût en temps, on peut considérer que Add, ScalMult et PMult ont un coût négligeable devant ceux de CMult et Rot, qui ont des coûts assez comparables.

§3.3. Batching

Le schéma CKKS doit manipuler des grands chiffrés ($N \in \{2^{14}, 2^{15}, 2^{16}\}$). Cette taille minimale des données n'est pas forcément idéale quand on veut manipuler des données plus petites. En contrepartie, CKKS dispose de propriétés SIMD (Single Instruction Multiple Data), c'est-à-dire que CKKS peut effectuer en même temps les mêmes opérations sur plusieurs données. On peut donc traiter plusieurs données en même temps, on appelle cela "batcher" ces données.

§4. La convolution

Notation 4.1: Pour $(n, m) \in \mathbb{N}^2$, on note $\mathcal{M}_n(\mathbb{R})$ l'ensemble des matrices carrées de taille n à coefficients réels et $\mathcal{M}_{n,m}(\mathbb{R})$ l'ensemble des matrices de format $n \times m$ à coefficients réels.
On notera alors :

- I_n la matrice identité de taille $n \times n$
- $0_{n,m}$ la matrice de format $n \times m$ dont tous les coefficients valent 0
- $J_{n,m}$ la matrice de format $n \times m$ dont tous les coefficients valent 1

Dans ce travail, une image traitée par un CNN sera représentée par des matrices de réels. Par exemple, pour le format RGB (Red, Green, Blue), les images sont représentées par trois matrices de même taille, une pour chaque couleur. Afin de chiffrer une image, on doit d'abord l'encoder : on encode les valeurs de la matrice ligne après ligne.

Définition 4.1 (Encodage lignes): Soit $n, m \in \mathbb{N}$
On définit la fonction d'encodage en lignes des matrices par :

$$\mathrm{Ecd}_l : \mathcal{M}_{n,m}(\mathbb{R}) \to \mathbb{R}^{nm} \to \mathcal{P}^{\lceil \frac{mn}{N} \rceil}$$
$$(a_{i,j}) \mapsto (b_{mi+j}) \mapsto \Big(\mathrm{Ecd}_{\mathrm{slot}}(b_0, b_1, ..., b_{N-1}); .....; \mathrm{Ecd}_{\mathrm{slot}}\Big(b_{\lfloor \frac{mn}{N} \rfloor N}, b_{\lfloor \frac{mn}{N} \rfloor N+1}, ..., b_{mn-1}, 0, ..., 0\Big)\Big)$$

Définition 4.2 (Conv2d): Soient $K \in \mathcal{M}_f(\mathbb{R})$ et $M \in \mathcal{M}_{h,w}(\mathbb{R})$.
Alors la convolution de M par le noyau K est définie par :
$M \circledast_{2\mathrm{d}} K = \big(c_{i,j}\big)_{\substack{(0 \le i \le h-f) \\ (0 \le j \le w-f)}} \in \Big(\mathcal{M}_{(h-f+1),(w-f+1)}\Big)(\mathbb{R})$
Avec

$$\forall (i,j) \in [\![0, h-f]\!] \times [\![0, w-f]\!], c_{i,j} = \sum_{n=0}^{f-1} \sum_{m=0}^{f-1} k_{n,m} m_{i+n,j+m}$$

Définition 4.3 (Conv2d avec stride et padding): Soient $K \in \mathcal{M}_f(\mathbb{R})$ et $M \in \mathcal{M}_{h,w}(\mathbb{R})$ et $s, p \in \mathbb{N}$
$M \circledast_{2\mathrm{d}}^{p,s} K = \big(c_{i,j}\big)_{\substack{(0 \le i \le \lfloor \frac{h+2p-f}{s} \rfloor) \\ (0 \le j \le \lfloor \frac{w+2p-f}{s} \rfloor)}} \in \mathcal{M}_{\lfloor \frac{h+2p-f}{s} \rfloor + 1, \lfloor \frac{w+2p-f}{s} \rfloor + 1}(\mathbb{R})$

$$\forall (i,j) \in \left[\!\!\left[0, \left\lfloor \frac{h+2p-f}{s} \right\rfloor\right]\!\!\right] \times \left[\!\!\left[0, \left\lfloor \frac{w+2p-f}{s} \right\rfloor\right]\!\!\right], c_{i,j} = \sum_{n=0}^{f-1} \sum_{m=0}^{f-1} k_{n,m} m_{si-p+n,sj-p+m}$$

En posant $m_{i,j} = 0$ lorsque ($i \notin [\![0, h-1]\!]$ ou $j \notin [\![0, w-1]\!]$)

Note 4.1: Cette deuxième définition est une généralisation de la première.

Exemple:

$$\begin{array}{|c|c|c|} \hline m_{0,0} & m_{0,1} & m_{0,2} \\ \hline m_{1,0} & m_{1,1} & m_{1,2} \\ \hline m_{2,0} & m_{2,1} & m_{2,2} \\ \hline \end{array} \circledast_{2\mathrm{d}}^{0,1} \begin{array}{|c|c|} \hline k_{0,0} & k_{0,1} \\ \hline k_{1,0} & k_{1,1} \\ \hline \end{array} = \begin{array}{|c|c|} \hline A & B \\ \hline C & D \\ \hline \end{array}$$

Avec $A = m_{0,0}k_{0,0} + m_{0,1}k_{0,1} + m_{1,0}k_{1,0} + m_{1,1}k_{1,1}$,et $D = m_{1,1}k_{0,0} + m_{1,2}k_{0,1} + m_{2,1}k_{1,0} + m_{2,2}k_{1,1}$

Comme précisé plus tôt, une image est représentée par plusieurs matrices de réels. On appelle ces différentes matrices des canaux. Une couche convolutionnelle d'un CNN prend en compte ces différents canaux, elle prend en entrée un nombre c_{in} de canaux, et retourne en sortie un nombre c_{out} de canaux, qui sont des combinaisons linéaires des convolutions des canaux d'entrée avec les noyaux correspondants.

Définition 4.4 (Convolution avec canaux): Soient $c_{\text{in}}, c_{\text{out}}, p, s \in \mathbb{N}$
Soient $K \in \left(\left(\mathcal{M}_f(\mathbb{R})\right)^{c_{\text{in}}}\right)^{c_{\text{out}}} = \left(K_{i,j}\right)_{\substack{(0\leq i\leq c_{\text{in}}-1)\\(0\leq j\leq c_{\text{out}}-1)}}$ et $M \in \left(\mathcal{M}_{h,w}(\mathbb{R})\right)^{c_{\text{in}}} = \left(M_i\right)_{0\leq i\leq c_{\text{in}}-1}$.
Alors la convolution des canaux de M sur les noyaux de K est définie comme :

$$M \circledast^{p,s} K = \left(C_j\right)_{0\leq j\leq c_{\text{out}}-1} \in \left(\mathcal{M}_{\lfloor\frac{h+2p-f}{s}\rfloor+1,\lfloor\frac{w+2p-f}{s}\rfloor+1}(\mathbb{R})\right)^{c_{\text{out}}}$$

Où

$$\forall j \in [\![0, c_{\text{out}}-1]\!], C_j = \sum_{i=0}^{c_{\text{in}}-1} M_i \circledast_{\text{2d}}^{p,s} K_{i,j}.$$

L'objectif de ce stage est alors d'effectuer de façon homomorphe et efficace cette opération de convolution sur des données chiffrées et des noyaux clairs.

§5. Produit matriciel homomorphe

Note 5.1: On appellera PC-MM (Plaintext-Ciphertext Matrix Multiplication) une multiplication entre une matrice claire et une matrice chiffrée ; qui prend en entrée une matrice claire $U \in \mathcal{M}_{d_1,d_2}(\mathbb{R})$ et des chiffrés représentant $M \in \mathcal{M}_{d_2,d_3}(\mathbb{R})$, la sortie de ce PC-MM est un ensemble de chiffrés représentant $U \cdot M \in \mathcal{M}_{d_1,d_3}(\mathbb{R})$.
Un PP-MM (Plaintext-Plaintext Matrix Multiplication) désigne une multiplication entre deux matrices claires.

Théorème 5.1 (réduction): Soient $d_1, d_2 \in \mathbb{N}$.
On suppose que $M \in \mathcal{M}_{d_2,N}(\mathbb{R})$ est donnée par les chiffrés de ses lignes, dans $\mathcal{R}^2_{Q,N}$, et que $U \in \mathcal{M}_{d_1,d_2}(\mathbb{R})$ est une matrice claire. Alors le PC-MM $U \cdot M$ se réduit à deux PP-MM de format $d_1 \times d_2 \times N$ dans $\frac{\mathbb{Z}}{Q\mathbb{Z}}$. La matrice à gauche du produit restant la même.

Grâce à des bibliothèques bien optimisées de multiplications de matrices claires (comme la librairie BLAS [6]), ce théorème nous donne une méthode bien plus rapide que l'état de l'art précédent pour effectuer les PC-MM. L'objectif de ce stage est d'utiliser cette technique pour accélérer l'étape de convolution des CNN, quitte à devoir batcher (c'est-à-dire traiter plusieurs images en même temps).

§6. Approche 1 : Im2col

§6.1. Définitions

Définition 6.1.1 (Encodage "Row Major" et "Column Major"): Pour $n, m, c \in \mathbb{N}$, on définit l'encodage Row Major d'une matrice (ou d'un vecteur de matrices) par :

- $\mathrm{RM}_{2d} : \mathcal{M}_{n,m}(\mathbb{R}) \to \mathcal{M}_{1,nm}(\mathbb{R}) \simeq \mathbb{R}^{nm}$

$$\left(a_{i,j}\right) \mapsto \begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,m-1} & \cdots\cdots & a_{n-1,0} & a_{n-1,1} & a_{n-1,m-1} \end{pmatrix}$$

- $\mathrm{RM}_{3d} : \left(\mathcal{M}_{n,m}(\mathbb{R})\right)^c \to \mathcal{M}_{c,nm}(\mathbb{R})$

$$\left(M_i\right)_{i \in [\![0,c-1]\!]} \mapsto \begin{pmatrix} \mathrm{RM}_{2d}(M_0) \\ \vdots \\ \mathrm{RM}_{2d}(M_{c-1}) \end{pmatrix}$$

De la même manière on définit l'encodage "Column Major", qu'on notera $\mathrm{CM}_{2\mathrm{d}}$ et $\mathrm{CM}_{3\mathrm{d}}$.

Définition 6.1.2 (Fonction de padding): On pose pour $i, j \in \mathbb{N}$:

$$\mathrm{Pad}_{i,j} : \mathcal{M}_{n,m}(\mathbb{R}) \to \mathcal{M}_{n+i,m+j}(\mathbb{R})$$
$$M \mapsto \left(\begin{array}{c|c} M & 0_{n,j} \\ \hline 0_{i,m} & 0_{i,j} \end{array}\right)$$

On étend ensuite cette définition aux vecteurs de matrices, posant pour $c \in \mathbb{N}$:

$$\mathrm{Pad}_{i,j} : \left(\mathcal{M}_{n,m}(\mathbb{R})\right)^c \to \left(\mathcal{M}_{n+i,m+j}(\mathbb{R})\right)^c$$
$$\left(M_k\right)_{k \in [\![0,c-1]\!]} \mapsto \left(\mathrm{Pad}_{i,j}(M_k)\right)_{k \in [\![0,c-1]\!]}$$

Par souci de clarté, on garde la même notation pour ces deux applications, la nature de l'entrée enlevant toute ambiguïté.

§6.2. Algorithme

Dans des situations sans chiffrement, l'idée de l'algorithme Im2col pour exprimer une convolution comme un produit matriciel est plutôt courante [7], on l'adapte ici dans le cas avec chiffrement homomorphe.

Exemple: Soient $M = \left(m_{i,j}\right) \in \mathcal{M}_{3,3}(\mathbb{R})$ et $K = \begin{pmatrix} k_1 & k_2 \\ k_3 & k_4 \end{pmatrix} \in \mathcal{M}_{2,2}(\mathbb{R})$, on veut retrouver les valeurs de $M \circledast_{2\mathrm{d}} K = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ en exprimant la convolution comme un produit matriciel.
On met K sous la forme $K \longrightarrow K' = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 \end{pmatrix}$. Le format im2col de la matrice M est construite de la manière suivante :

$$M = \begin{array}{|c|c|c|} \hline m_{0,0} & m_{0,1} & m_{0,2} \\ \hline m_{1,0} & m_{1,1} & m_{1,2} \\ \hline m_{2,0} & m_{2,1} & m_{2,2} \\ \hline \end{array} \longrightarrow M' = \begin{array}{|c|c|c|c|} \hline m_{0,0} & m_{0,1} & m_{1,0} & m_{1,1} \\ \hline m_{0,1} & m_{0,2} & m_{1,1} & m_{1,2} \\ \hline m_{1,0} & m_{1,1} & m_{2,0} & m_{2,1} \\ \hline m_{1,1} & m_{1,2} & m_{2,1} & m_{2,2} \\ \hline \end{array}$$

On a alors

$$K' \times M' = \begin{array}{|c|c|c|c|} \hline A & B & C & D \\ \hline \end{array}$$

Considérons ici que $N \geq dhw$ où d est le nombre d'images que l'on prend en entrée. L'algorithme ici manipule c_{in} chiffrés, un pour chaque canal d'entrée. Chaque chiffré représente un certain canal des d images que l'on batch, stockées ligne par ligne et les unes derrière les autres.

On veut d'abord récupérer la matrice au format Im2col [7]. Pour cela on considère chaque fenêtre de format $(h-f+1)\times(w-f+1)$ qui tiennent dans une image, une fenêtre de ce format correspond à l'ensemble des coordonnées de l'image qui peuvent se retrouver en face d'une même valeur du noyau. Une telle fenêtre contient donc les valeurs d'une ligne de la matrice au format Im2col. On exploite cette idée dans l'algorithme suivant afin de récupérer la matrice au format Im2col en représentation ligne, afin de pouvoir utiliser la technique PC-MM.

On pose $K \in \left(\mathcal{M}_f(\mathbb{R})^{c_{\text{in}}}\right)^{c_{\text{out}}}$ le noyau (de la même forme qu'avant) et $M = (M_0, ..., M_{d-1}) \in \left(\mathcal{M}_{h,w}(\mathbb{R})^{c_{\text{in}}}\right)^d$ un vecteur d'images qu'on veut batcher.

$\underline{\text{Im2col}}(K \in \left(\mathcal{M}_f(\mathbb{R})^{c_{\text{in}}}\right)^{c_{\text{out}}}, (\text{ct}_i)_{i\in[\![0,c_{\text{in}}-1]\!]})$:

For $i \in [\![0, c_{\text{out}} - 1]\!]$ **do :**
  $\text{output}_i \leftarrow 0_{1,N}$
End For
$\text{mask} \leftarrow \text{Ecd}\big(\text{RM}_{2\text{d}}\big(\text{RM}_{3\text{d}}\big(\text{Pad}_{f-1,f-1}\big(J_{h-f+1,w-f+1}, ..., J_{h-f+1,w-f+1}\big)\big)\big)\big)$
For $n \in [\![0, c_{\text{in}} - 1]\!]$ **do:** \\mise en im2col du n-ème channel
  For $i \in [\![0, f-1]\!]$ **do:**
    For $j \in [\![0, f-1]\!]$ **do:**
      $\text{ct}'_{n,(fi+j)} \leftarrow \text{Rot}_{wi+j}(\text{ct}_n)$
      $\text{ct}'_{n,(fi+j)} \leftarrow \text{ct}'_{n,(fi+j)} \times \text{mask}$
    End For
  End For
$\text{res} \leftarrow \begin{pmatrix} \text{RM}_{3d}(K_0) & ... & \text{RM}_{3d}\big(K_{c_{\text{in}}-1}\big) \end{pmatrix} \times \begin{pmatrix} \text{ct}'_{0,0} \\ \vdots \\ \text{ct}'_{0,f^2-1} \\ \vdots \\ \text{ct}'_{c_{\text{in}}-1,0} \\ \vdots \\ \text{ct}'_{c_{\text{in}}-1,f^2-1} \end{pmatrix}$ \\PC-MM $c_{\text{out}} \times c_{\text{in}} f^2 \times hw$
return res

Théorème 6.2.1 (sortie): Si l'algorithme prend en entrée $K \in \left(\mathcal{M}_f(\mathbb{R})^{c_{\text{in}}}\right)^{c_{\text{out}}}$ et $(\text{ct}_i)_{i\in[\![0,c_{\text{in}}-1]\!]}$ avec

$$\forall i \in [\![0, c_{\text{in}} - 1]\!], \text{ct}_i = \text{Enc}\left(s_k, \text{Ecd}_l\left(\begin{array}{c} M_{0,i} \\ \hline M_{1,i} \\ \hline \vdots \\ \hline M_{d-1,i} \end{array}\right)\right) \text{ pour } (M_{n,i}) \in \left(\mathcal{M}_{h,w}(\mathbb{R})^{c_{\text{in}}}\right)^d,$$

alors sa sortie sera : $(\text{ct}''_i)_{i\in[\![0,c_{\text{out}}-1]\!]}$
avec

$$\forall i \in [\![0, c_{\text{out}} - 1]\!],$$
$$\text{ct}''_i = \text{Enc}\Big(\text{Ecd}_l\Big(\text{RM}_{3\text{d}}\Big(\text{Pad}_{f-1,f-1}\Big(\big(M_0 \circledast^{0,1} K\big)_i, \big(M_1 \circledast^{0,1} K\big)_i, ..., \big(M_{d-1} \circledast^{0,1} K\big)_i\Big)\Big)\Big)\Big)$$

Théorème 6.2.2 (Coût de l'algorithme): Dans cet algorithme, on utilise :
- $c_{\text{in}} f^2$ rotations
- Un PC-MM de format $c_{\text{out}} \times c_{\text{in}} f^2 \times N$ (qui se réduisent chacun à 2 PP-MM de même format)
- Perte de deux niveaux

Note 6.2.1: Le format en sortie n'est pas le même que le format en entrée, pour pouvoir envoyer ces données à la prochaine couche du CNN, il faut réorganiser les données, pour cela on peut utiliser l'outil Hermes [8], dont l'utilisation est très coûteuse (environ le même coût qu'un Bootstrapp). De plus, Hermès doit être utilisé pendant un Bootstrap, on fait alors Hermes et Bootstrapp pour le coût d'environ deux Bootstrapp.

§7. Approche 2 : Multiplication polynomiale

Cette approche nécessite de traiter N images en même temps, ce qui est une contrainte non négligeable dans la mesure où $N \in \{2^{14}, 2^{15}\}$. Dans toute cette partie on supposera donc $d = N$.

§7.1. Conv2d comme une mutiplication polynomiale

La convolution 2d peut-être représentée par un produit polynomial [9].

Théorème 7.1.1: Soit $M = (m_{i,j}) \in \mathcal{M}_{h,w}(\mathbb{R})$ et $K = (k_{i,j}) \in \mathcal{M}_f(\mathbb{R})$.
On effectue le produit polynomial suivant dans $\frac{\mathbb{R}[X]}{(X^{hw}-1)}$, et on note :

$$P = \sum_{i=0}^{hw-1} P_i X^i = \left(\sum_{i=0}^{f-1} \sum_{j=0}^{f-1} k_{i,j} X^{hw-(iw+j)} \right) \times \left(\sum_{i=0}^{h-1} \sum_{j=0}^{w-1} m_{i,j} X^{iw+j} \right)$$

On a alors $\forall i \in [\![0, h-f]\!], \forall j \in [\![0, w-f]\!], (M \circledast_{\text{2d}} K)_{i,j} = P_{iw+j}$

Pour un noyau $K \in \mathcal{M}_f(\mathbb{R})$, notons P_K le polynôme avec les coefficients de K présenté ci-dessus. Pour une image $M \in \mathcal{M}_{h,w}(\mathbb{R})$, notons P_M le polynôme avec les coefficients de M présenté ci-dessus.

Preuve: Soit $i \in [\![0, h-f]\!]$ et $j \in [\![0, w-f]\!]$

$$\begin{aligned}
P_{wi+j} &= \sum_{\substack{i_1,j_1,i_2,j_2 \text{ tq} \\ i_1 w + j_1 + hw - i_2 w - j_2 = iw+j \bmod (hw)}} m_{i_1,j_1} k_{i_2,j_2} \\
&= \sum_{\substack{i_1,j_1,i_2,j_2 \text{ tq} \\ (i_1-i_2)w + j_1 - j_2 = iw+j}} m_{i_1,j_1} k_{i_2,j_2} \\
&= \sum_{\substack{i_1,j_1,i_2,j_2 \text{ tq} \\ i_1 - i_2 = i \text{ et } j_1 - j_2 = j}} m_{i_1,j_1} k_{i_2,j_2} \\
&= \sum_{i_2=0}^{f-1} \sum_{j_2=0}^{f-1} m_{i+i_2,j+j_2} k_{i_2,j_2} \\
&= (M \circledast_{\text{2d}} K)_{i,j}
\end{aligned}$$

■

Grâce à un produit polynomial dans l'anneau $\frac{\mathbb{R}[X]}{(X^{hw}-1)}$, on peut alors retrouver les coefficients de la convolution 2d, il suffit "d'oublier" les coefficients en trop pour ne garder que ceux qui nous intéressent.

§7.2. La multiplication polynomiale comme un produit matriciel

Définition 7.2.1 (Matrice de multiplication polynomiale): Dans $\frac{\mathbb{R}[X]}{(X^{hw}-1)}$, pour un polynôme $P = \sum_{i=0}^{hw-1} P_i X^i$ la matrice de l'application : $Q \mapsto PQ$ dans la base $\left(1, X, ..., X^{hw-1}\right)$ est :

$$M(P) = \begin{pmatrix} P_0 & P_{hw-1} & \cdots & \cdots & P_2 & P_1 \\ P_1 & P_0 & P_{hw-1} & \cdots & P_3 & P_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ P_{hw-2} & P_{hw-3} & \cdots & \cdots & P_0 & P_{hw-1} \\ P_{hw-1} & P_{hw-2} & \cdots & \cdots & P_1 & P_0 \end{pmatrix}$$

Elle est de format $hw \times hw$.

On sait donc exprimer la convolution 2d comme un produit polynomial dans un certain anneau, qu'on peut exprimer lui-même comme un produit matrice-vecteur. En batchant suffisamment d'images (ici N images, puisqu'un vecteur correspond à un canal d'une image), on peut donc exprimer la convolution 2d de toutes ces images comme un produit matrice-matrice, et donc utiliser la technique PC-MM pour se ramener à des calculs clairs.

Plus précisement, on chiffre un même pixel des N images dans un seul chiffré, un channel des N images est représenté par hw chiffrés : un pour chaque pixel :

$$\forall i \in [\![0, h-1]\!], \forall j \in [\![0, w-1]\!], \mathrm{ct}_{i,j} = \left(a_{i,j}, b_{i,j}\right) = \mathrm{Enc}\Big(s_k, \mathrm{Ecd}\Big(M^0_{i,j}, ..., M^{N-1}_{i,j}\Big)\Big)$$

Ces chiffrés représentent les lignes de la matrice chiffrée, que l'on note M. Le PC-MM $M(P_K) \cdot M$ effectue alors l'opération de convolution sur les N images qui forment la matrice M, et se réduit à deux PP-MM où la matrice à gauche reste la même : $M(P_K) \cdot A$ et $M(P_K) \cdot B$.

D'après la définition précédente, ces calculs reviennent à considérer chaque colonne de A (resp. B) comme un polynôme et à calculer son produit avec P_K. Chacun de ces PP-MM se réduit donc à N produits polynomiaux (un pour chaque colonne). Donc le PC-MM se réduit à $2N$ produits polynomiaux clairs.

§7.3. Encodage coordonnée par coordonnée

Définition 7.3.1 (enc coord / coord): On définit l'encodage coordonnée par coordonnée comme :

$$\begin{aligned} \mathrm{Enc}_{\mathrm{coord}} : \mathcal{M}_{h,w}(\mathbb{R})^N &\to \mathcal{C}^{hw} \\ \left(M^j\right)_{j \in [\![0,N-1]\!]} &\mapsto \mathrm{Enc}\left(\mathrm{Ecd}_l\Big(\mathrm{CM}_{2\mathrm{d}}(M^{0^T}) \ \ldots \ \mathrm{CM}_{2\mathrm{d}}\Big(M^{(N-1)^T}\Big)\Big)\right) \end{aligned}$$

Dans ce format de chiffrement des images, on a hw chiffrés qui représentent chacun un seul pixel de N images différentes.

§7.4. Algorithme

$\underline{\text{MULT\_POLY}}\left(\left(K_{n,m}\right) \in \left(\mathcal{M}_f(\mathbb{R})^{c_{\text{in}}}\right)^{c_{\text{out}}}, \left(\left(\text{ct}_i^m\right)_{i\in[\![0,hw-1]\!]}\right)_{m\in[\![0,c_{\text{in}}-1]\!]} \in \left(\mathcal{C}^{hw}\right)^{c_{\text{in}}}\right)$:

1 $\begin{pmatrix} {\text{ct}'_0}^0 \\ \vdots \\ {\text{ct}'_{hw-1}}^0 \\ \vdots \\ {\text{ct}'_0}^{c_{\text{out}}-1} \\ \vdots \\ {\text{ct}'_{hw-1}}^{c_{\text{out}}-1} \end{pmatrix} \longleftarrow \begin{pmatrix} M\left(P_{K_{0,0}}\right) & \dots & M\left(P_{K_{0,c_{\text{in}}-1}}\right) \\ \vdots & \vdots & \vdots \\ M\left(P_{K_{c_{\text{out}}-1,0}}\right) & \dots & M\left(P_{K_{c_{\text{out}}-1,c_{\text{in}}-1}}\right) \end{pmatrix} \times \begin{pmatrix} {\text{ct}'_0}^0 \\ \vdots \\ {\text{ct}'_{hw-1}}^0 \\ \vdots \\ {\text{ct}'_0}^{c_{\text{in}}-1} \\ \vdots \\ {\text{ct}'_{hw-1}}^{c_{\text{in}}-1} \end{pmatrix}$

2 On groupe tous les canaux en un seul PC-MM

3 **Return** $\left(\left({\text{ct}'_{iw+j}}^n\right)_{\substack{i\in[\![0,h-f]\!]\\ j\in[\![0,w-f]\!]}}\right)_{n\in[\![0,c_{\text{out}}-1]\!]}$

Théorème 7.4.1 (Résultat de l'algorithme): Si l'algorithme prend en entrée un noyau K et un l'encodage coordonnée par coordonnée de N images (avec leurs différents canaux), alors il renvoie en sortie l'encodage coordonnée par coordonnée de la convolution de ces N images par le noyau K (avec leurs différents canaux).

Théorème 7.4.2 (Coût de l'algorithme): Dans cet algorithme, on utilise :
- Aucune rotation
- Un PC-MM de format $c_{\text{out}}hw \times c_{\text{in}}hw \times N$
- Perte d'un niveau

Note 7.4.1: Ici, le format de sortie est le même que le format d'entrée, on peut alors directement récupérer les données de sortie d'une couche convolutionnelle et les donner en entrée de la couche suivante, ce qui nous évite du temps de calcul pour la réorganisation des données.

§7.5. Approche 2bis

Comme dit précédemment, on peut réduire le PC-MM ici à des produits polynomiaux. On peut d'ailleurs voir ces produits et sommes de polynômes comme un produit de matrices à coefficients dans $\frac{\frac{\mathbb{Z}}{q\mathbb{Z}}[X]}{(X^{hw}-1)}$:

$$\begin{pmatrix} P_{K_{0,0}} & \dots & P_{K_{0,c_{\text{in}}-1}} \\ \vdots & \vdots & \vdots \\ P_{K_{c_{\text{out}}-1,0}} & \dots & P_{K_{c_{\text{out}}-1,c_{\text{in}}-1}} \end{pmatrix} \times \begin{pmatrix} P_{M_{0,0}} & \dots & P_{M_{0,d-1}} \\ \vdots & \vdots & \vdots \\ P_{M_{c_{\text{in}}-1,0}} & \dots & P_{M_{c_{\text{in}}-1,d-1}} \end{pmatrix}$$

Pour calculer $K \cdot A$ où K et A sont des matrices de polynômes de $\frac{\frac{\mathbb{Z}}{q\mathbb{Z}}[X]}{(X^{hw}-1)}$, on peut se ramener à calculer $K(\omega^i) \times A(\omega^i)$ pour ω une racine primitive de l'unité modulo q, en supposant $q = 1 \bmod hw$. Puis on reconstruit $K \times A$ à partir de ces matrices. Passer d'une matrice en ses valeurs en les ω^i requiert une NTT par coefficient, de degré hw, reconstruire une matrice à partir de ses valeurs en les ω^i requiert une iNTT par coefficient.

Théorème 7.5.1 (Coût de l'algorithme): Dans cette nouvelle approche, on utilise :
- Aucune rotation
- Aucune multiplication de chiffrés
- $2Nc_{\text{in}}$ NTT de degré hw
- $2Nc_{\text{out}}$ iNTT de degré hw
- $2hw$ PP-MM de format $c_{\text{out}} \times c_{\text{in}} \times N$

§8. Résultats

Dans cette partie, on estime en pratique l'efficacité de nos deux algorithmes. Sans les implémenter complètement, on mesure le temps d'exécution des opérations les plus coûteuses pour estimer le temps d'exécution de l'algorithme complet.

- Les temps des NTT sont mesurés en single-thread CPU sur un Intel(R) Xeon(R) Gold 6242 2.80GHz.
- Les temps des rotations de chiffrés sont mesurés en utilisant la bibliothèque HEaaN [10] en single-thread CPU sur un Intel(R) Xeon(R) 6242 CPU 2.80GHz.
- Les temps des produits matriciels sont mesurés en utilisant la librairie BLAS [6] en single-thread CPU sur un processeur Intel(R) Core(TM) i5 2.40GHz.

§8.1. Valeurs classiques pour un CNN ResNet18

Un CNN de type ResNet18 possède plusieurs sortes de couches convolutionnelles :

couche	format in	format out	c_{in}	c_{out}	f	s	p
1	224×224	112×112	3	64	7	2	3
2	56×56	56×56	64	64	3	1	1
3	56×56	28×28	64	128	3	2	1
4	28×28	28×28	128	128	3	1	1
5	28×28	14×14	128	256	3	2	1
6	14×14	14×14	256	256	3	1	1
7	14×14	7×7	256	512	3	2	1
8	7×7	7×7	512	512	3	1	1

Table 1: Valeurs usuelles des couches convolutionnelles d'un ResNet18

§8.2. Temps d'évaluation homomorphe de ResNet18 : méthode im2col

couche	nb Rot	temps Rot (ms)	format PC-MM	Temps d'un PP-MM (ms)	Nombre d'images qu'on peut batcher	Temps total (ms)	Temps par image (ms)
1	147	3969	$64 \times 147 \times N$	4.969	< 1	4167	
2	576	15552	$64 \times 576 \times N$	18.460	10	16290	1629
3	576	15552	$128 \times 576 \times N$	30.277	10	16763	1676
4	1152	31104	$128 \times 1152 \times N$	61.800	41	33576	818
5	1152	31104	$256 \times 1152 \times N$	116.078	41	35747	871
6	2304	62208	$256 \times 2304 \times N$	234.926	128	71605	559
7	2304	62208	$512 \times 2304 \times N$	458.284	128	80539	629
8	4608	124410	$512 \times 4608 \times N$	954.870	404	162610	402
Total							15036

Table 2: Temps d'exécution des différents PP-MM d'un Resnet18 selon la taille d'un chiffré, pour l'approche Im2col. (Au niveau 4 avec $N = 2^{16}$)

§8.3. Temps d'évaluation homomorphe de ResNet18 : méthode par la mutliplication polynomiale

Couche	nb NTT	degré NTT	temps NTT (ms)	nb PP-MM	format PP-MM	temps d'un PP-MM (ms)	Nb images batchée	temps total (ms)	temps par image (ms)
1	2195456	50176	790364	50176	$64 \times 3 \times N$	0.751	2^{15}	4705464	143.60
2	4194304	3136	14680	3136	$64 \times 64 \times N$	2.573	2^{15}	234778	7.16
3	6291456	3136	22020	3136	$128 \times 64 \times N$	4.322	2^{15}	381176	11.63
4	8388608	784	29360	784	$128 \times 128 \times N$	7.480	2^{15}	264087	8.06
5	12582912	784	44040	784	$256 \times 128 \times N$	14.173	2^{15}	442433	13.50
6	16777216	196	14763	196	$256 \times 256 \times N$	26.735	2^{15}	178620	5.45
7	25165824	196	22145	196	$512 \times 256 \times N$	52.155	2^{15}	315177	9.62
8	33554432	49	9730	49	$512 \times 512 \times N$	103.350	2^{15}	149937	4.58
Total									261.27

Table 3: Temps d'éxécution des différentes couches de ResNet18 avec la méthode polynomiale (pour $N = 2^{16}$, au niveau 4)

Bibliography

[1] Y. Bae, J. H. Cheon, G. Hanrot, J. H. Park, and D. Stehlé, "Plaintext-Ciphertext Matrix Multiplication and FHE Bootstrapping: Fast and Fused," 2024.

[2] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, M. Mitzenmacher, Ed., ACM, 2009, pp. 169–178. doi: 10.1145/1536414.1536440.

[3] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic Encryption for Arithmetic of Approximate Numbers." Accessed: Jun. 03, 2024. [Online]. Available: https://eprint.iacr.org/2016/421

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[5] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of Learning with Errors." [Online]. Available: https://eprint.iacr.org/2015/046

[6] L. S. Blackford *et al.*, "An updated set of basic linear algebra subprograms (BLAS)," *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 135–151, 2002.

[7] K. Chellapilla, S. Puri, and P. Simard, "High Performance Convolutional Neural Networks for Document Processing," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, G. Lorette, Ed., La Baule (France): Suvisoft, Oct. 2006. [Online]. Available: https://inria.hal.science/inria-00112631

[8] Y. Bae, J. H. Cheon, J. Kim, J. H. Park, and D. Stehlé, "HERMES: Efficient Ring Packing using MLWE Ciphertexts and Application to Transciphering." [Online]. Available: https://eprint.iacr.org/2023/1244

[9] J. H. Ju, J. Park, J. Kim, D. Kim, and J. H. Ahn, "NeuJeans: Private Neural Network Inference with Joint Optimization of Convolution and Bootstrapping." [Online]. Available: https://arxiv.org/abs/2312.04356

[10] "HEaaN librairy." [Online]. Available: https://heaan.it/

[11] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption." [Online]. Available: https://eprint.iacr.org/2012/144

[12] L. Rovida and A. Leporati, "Encrypted Image Classification with Low Memory Footprint Using Fully Homomorphic Encryption," *International Journal of Neural Systems*, vol. 34, no. 5, p. 2450025–2450026, May 2024, doi: 10.1142/S0129065724500254.

[13] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption," *CoRR*, 2021, [Online]. Available: https://arxiv.org/abs/2104.03152