國立清華大學 電機工程學系

實作專題研究成果摘要

Hardware Design for U-Net Accelerator in Stable Diffusion

Stable Diffusion 中的 U-Net 加速器硬體實作

專題領域：系統領域

組　　別：B354

指導教授：呂仁碩

組員姓名：許博翔、歐曄

研究期間：2023年1月3日 至 2023年11月29日止，共11個月

Abstract

This study aims to design a method to accelerate the speed of Stable Diffusion in generating images from text. Stable Diffusion, as a recently introduced image generative model, has gained attention for its powerful capabilities of image synthesis. However, despite its capabilities, there has been limited adoption of Stable Diffusion in services or commercial applications. We attribute this to the model's suboptimal performance in terms of computational speed. Therefore, we believe that finding methods to accelerate image generation with Stable Diffusion could lead to better development and utilization in the future.

In this research, we segmented Stable Diffusion into different blocks based on functionality. By measuring the time each block takes in image generation, we identified a performance bottleneck in Stable Diffusion, namely, U-Net. Through analyzing the architecture and characteristics of U-Net, we proposed two hardware design methods to accelerate its computational speed.

Firstly, we designed the Flow Process Unit as the basic unit for U-Net's convolution operation, reducing the number of clock cycles needed for a single-layer U-Net to complete convolution. Additionally, by adjusting the datapath, each layer of U-Net can send the already computed partial results to the next layer during convolution. When the next layer collects enough data, it can start its computation without waiting for the previous layer's computation to fully finish, achieving Pipelining acceleration.

Finally, through theoretical calculations, we demonstrated that our design hardware architecture can execute U-Net computations with fewer clock cycles compared to the traditional combination of CPU and C++. This serves as evidence of the successful acceleration of Stable Diffusion image generation speed through our design.

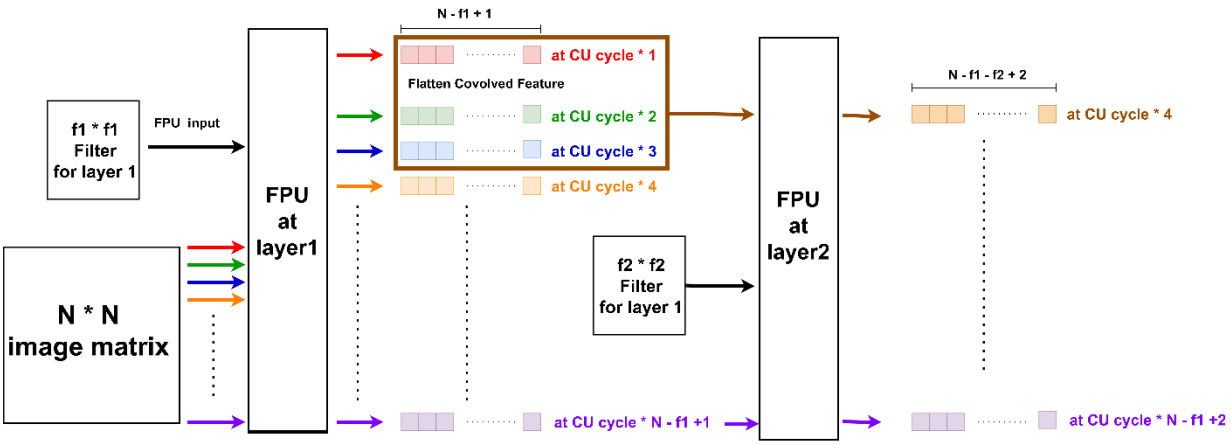

Figure 1. Simplified illustration for our design

中文摘要

本研究旨在設計出能加速 Stable Diffusion 文字生成圖片速度的方法，Stable Diffusion 做為近期推出的圖片生成模型，其強大的圖片生成功能是人們注目的焦點，但是到目前依舊鮮少運用 Stable Diffusion 的服務或商業上的使用出現，我們認為其原因在於 Stable Diffusion 在運算速度上表現不夠優異，所以我們認為如果能找到方法加速其生成圖片的速度的話，能夠使其在未來能有更好的發展及運用。

在這份研究中，我們將 Stable Diffusion 依照功能劃分成不同區塊，並透過測量各區塊在圖片生成所花的時間長度發現了 Stable Diffusion 中的效能瓶頸，也就是 U-Net。藉由分析 U-Net 的架構與特性，我們提出兩個硬體設計解決方向來加速 U-Net 的運算速度。

首先我們設計出 Flow Process Unit 做為 U-Net 進行卷積運算的基本單位，使單層 U-Net 完成卷積所需的 clock cycle 數降低，並且，我們透過調整 datapath，使得 U-Net 的每一層在進行卷積時可以先將已運算完的部分結果先送至下一層，當下一層收集到足夠的資料後即可開始該層的運算，不必等待上一層的運算全部結束後才開始，借此達到 Pipelining 加速效果。

最後，透過理論計算，我們證明了我們所設計的硬體架構能比傳統 CPU 與 C++的組合花費更少的 clock cycle 執行 U-Net 的運算，以此證明我們的設計具有成功加速 Stable Diffusion 圖片生成速度之效果。

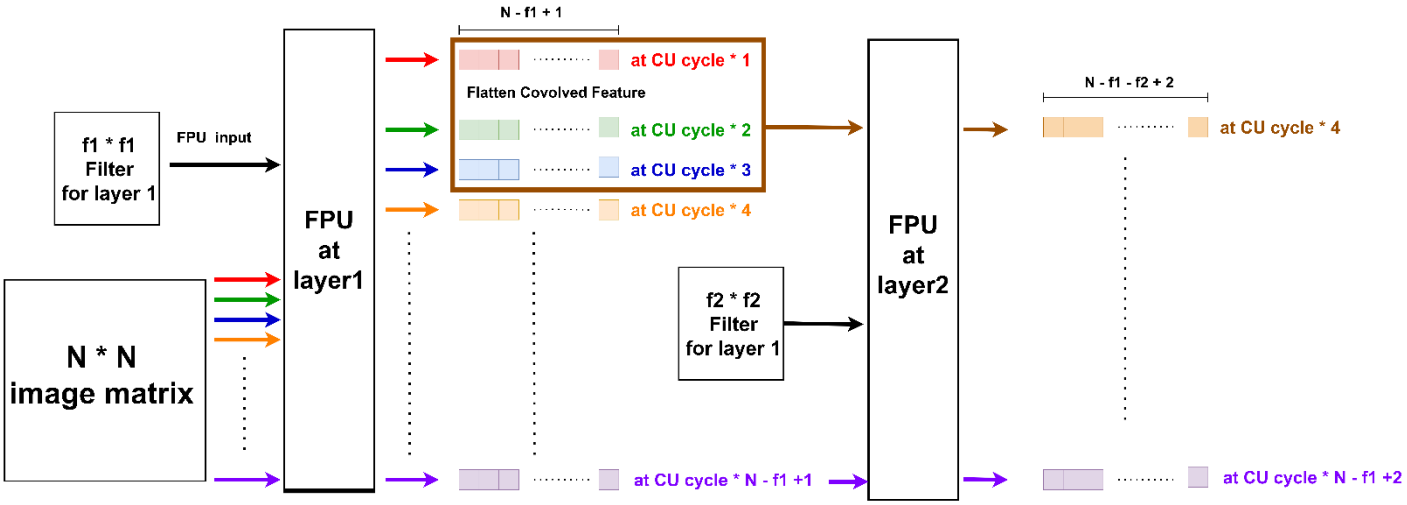

Figure 2. 我們設計的簡易說明圖

透過我們設計的 Flow Process Unit (FPU)以及 Pipelining 技巧的運用，能使 U-Net 在運算時花費更少的 clock cycle

1. Background and Motivation

2022年8月，一個基於潛在擴散模型(latent diffusion model; LDM)的圖片生成模型(Image generative model) — Stable Diffusion [1]出現在了世人眼中，這款模型強大的功能大大改變了人們對於「圖片生成」的看法。它的功能包括：根據輸入的文字描述生成細緻的圖像、進行圖像修復或擴展，也可以根據提示詞對現有圖片進行微調。其中，最受關注且被廣泛應用的是它的文字生成圖片("txt2img")功能。

其中值得一提的是，Stable Diffusion 的原始碼和模型權重已經分別公開在 GitHub 和 Hugging Face 上，同時 Stable Diffusion 的架構允許其他人在基礎的權重上進行訓練。任何擁有支援 CUDA 核心函數之操作系統的用戶都可以在自己的硬體上使用此模型生成圖片。

但即使是在模型開源的情況下，我們很驚訝地發現目前還沒有許多服務充分運用 Stable Diffusion。我們認為主要原因是對於一般用戶而言，使用 Stable Diffusion 生成一張高品質的圖片所需要的時間依舊過長，同時硬體要求也不低，故無法充分利用其價值。

2. Purpose

我們思考，若能夠加速 Stable Diffusion 在完成文字生成圖片這項任務的速度，將有助於 Stable Diffusion 更廣泛地應用於各種服務上。但是目前不乏從軟體角度下手去加速 Stanble Diffusion 的方法提出，但 Stable Diffusion 的運用依然稀少，因此我們決定換個出發點，計畫透過硬體設計的角度，提出實際可行的加速方法，以提高 Stable Diffusion 的運算效率。

為了實現加速這個目標，我們首先得了解 Stable Diffusion 的架構。我們按照工作目的將 Stable Diffusion 分為以下三個區塊：變分自編碼器（Variational AutoEncoder; VAE）、U-Net 和輸入編碼器。其中，VAE 的任務是將在 pixel space 的圖像轉換到低維潛在空間（Latent space），並在最後把在 Latent space 生成的圖像轉換回 Pixel space。輸入編碼器則負責將輸入的提示詞或圖像轉換為調節 U-Net 去噪過程的參數。U-Net 透過輸入編碼器的輸出預測噪聲，並將其從 Latent space 的圖像中去除。

通過實際測量上述三區塊在完成「文字生成圖片」這項任務中所佔的時間發現，VAE 和輸入編碼器的總執行時間僅約佔整個任務時間的不到1%。相對的，99%以上的時間花費在 U-Net 上（參見下方的分析表 Table 1.）。因此，**如果能夠提高 U-Net 的計**

算速度，將有效加速 Stable Diffusion 完成文字生成圖片的速度。

Prompts	a girl sitting by lake	a girl sitting by lake	a girl sitting by lake
Sample Number	80	40	80
Height	512	512	256
Width	512	512	512
Latent Channels	4	4	4
Downsampling Factor	8	8	8
Text Encode Time (s)	0.008319	0.0095046	0.010333
U-Net Time (s)	22.9847	11.041625	8.19529
VAE Time (s)	0.0049999	0.00511	0.0034203
Total Time (s)	22.9980189	11.0562396	8.2090433
Text Encode Time (%)	0.03619364186	0.08607972106	0.1260846169
U-Net Time (%)	99.94208675	99.86781582	99.8324616
VAE Time (%)	0.02174056827	0.04621824585	0.04166502569

Table 1. 上述三區塊在透過同樣的 prompt 及不同 sample number 和 image 大小在執行文字生成圖片任務中分別所佔的時間長度(青色區塊)以及百分比(藍色區塊)，可以明顯發現不管 sample number 以及 image 大小的變動，U-Net 所佔的時間都達99%以上。測試平台皆為 intel i7-13700 + Nvidia RTX4080 + DDR5 32GB

U-Net[2]是一種卷積神經網絡(CNN)，由一個收縮路徑（contracting path）和一個擴展路徑（expansive path）組成（見下示意圖 Figure 1.）。

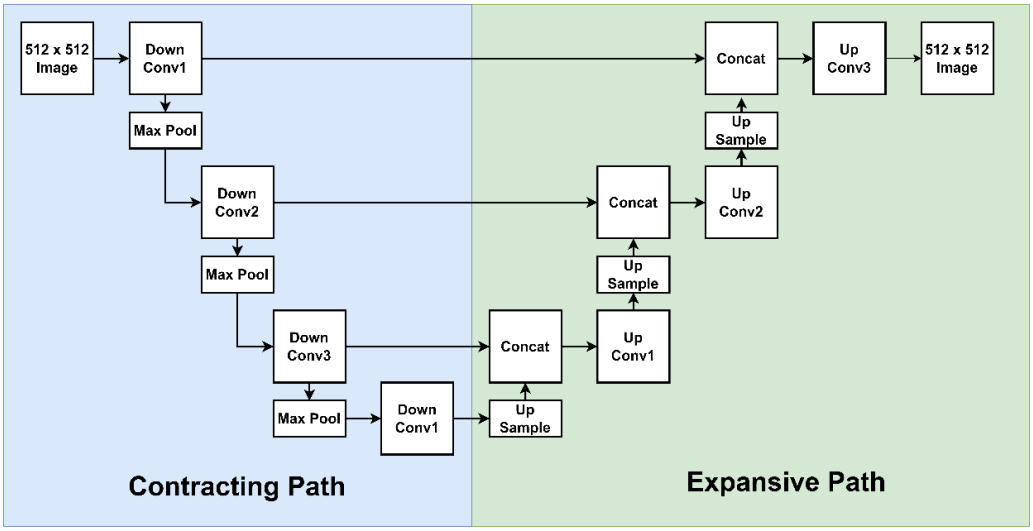

Figure 1. U-Net 簡易架構圖

其中，收縮路徑是一張典型的卷積網絡，包括卷積的重複應用，每個卷積之後都有一個線性整流函數單元（ReLU）和一個最大匯集作業（Max Pooling Operation）。擴張路徑則通過連續的上卷積和與來自收縮路徑的高解析度特徵相連接，以結合特徵與空間信息，即是說 **U-Net 包含大量的卷積計算，如果能夠降低 U-Net 運算卷積的時間，即可大大加快 Stable Diffusion 生成圖片的速度**。

3. Method

3.1 Flow Process Unit

我們設計了一種名為 Flow-Processing Unit (FPU)的硬體模組做為卷積運算的基本單位，以下設欲卷積的圖像寬度為 N，Filter 寬度為 f，對應的 Convolved feature 即為 $(N-f+1)*(N-f+1)$的矩陣。

FPU 的輸入為整個 filter 以及圖像中與 filter 高度同高的子區塊(即$f*N$的矩陣)，其目的為在一個 CU cycle (Convolution Unit 完成運算所需的 clock cycle 數，細節留置後面討論)內處理完將 Filter 從圖像上最左邊滑動至最右邊的運算，即生成出一個$1*N$的 Flatten Convolved Feature 矩陣(如下 Figure 2.所示)

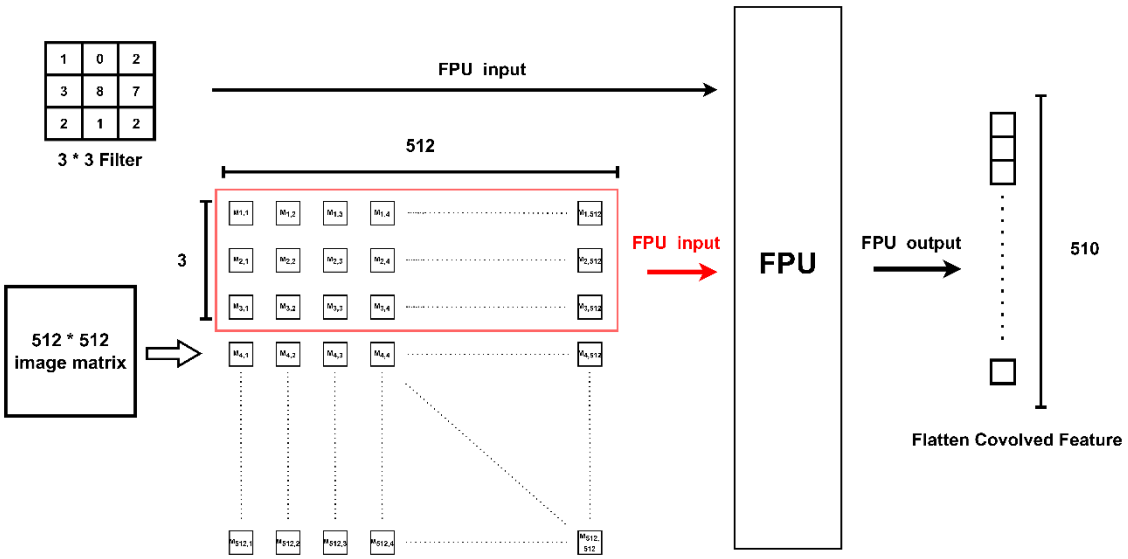

Figure 2. FPU 的 I/O 說明，(以512*512的 image 以及3*3的 filter 為例)，FPU 會吃進完整的 Filter 以及 Image 中3*512的資料進去，並輸出$1*(512-3+1\ =\ 510)$的 Convolved Feature

FPU 內部由$(N-f+1)$個 Convolution Unit (CU)組成，每個 CU 負責在一個 CU cycles 內將與 Filter 同大小的資料與 Filter 做 Element-wise Multiplication 並輸出其乘積(如下 figure 3.所示)，所以$(N-f+1)$個 CU 可使得 FPU 在一個 CU cycle 中完成一整個

橫排的卷積運算(如下 figure 3.所示)

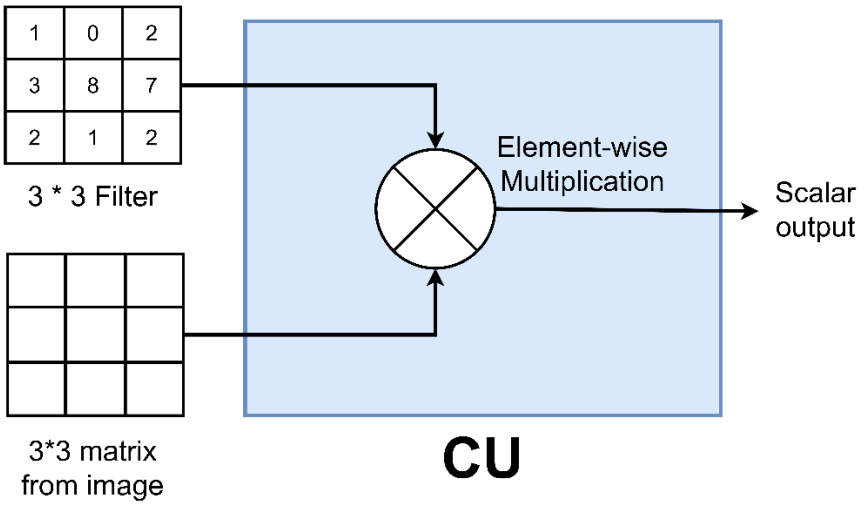

Figure 3. CU 的 I/O 說明，即將輸入的兩個矩陣做 Element-wise Multiplication 並輸出其乘積總和

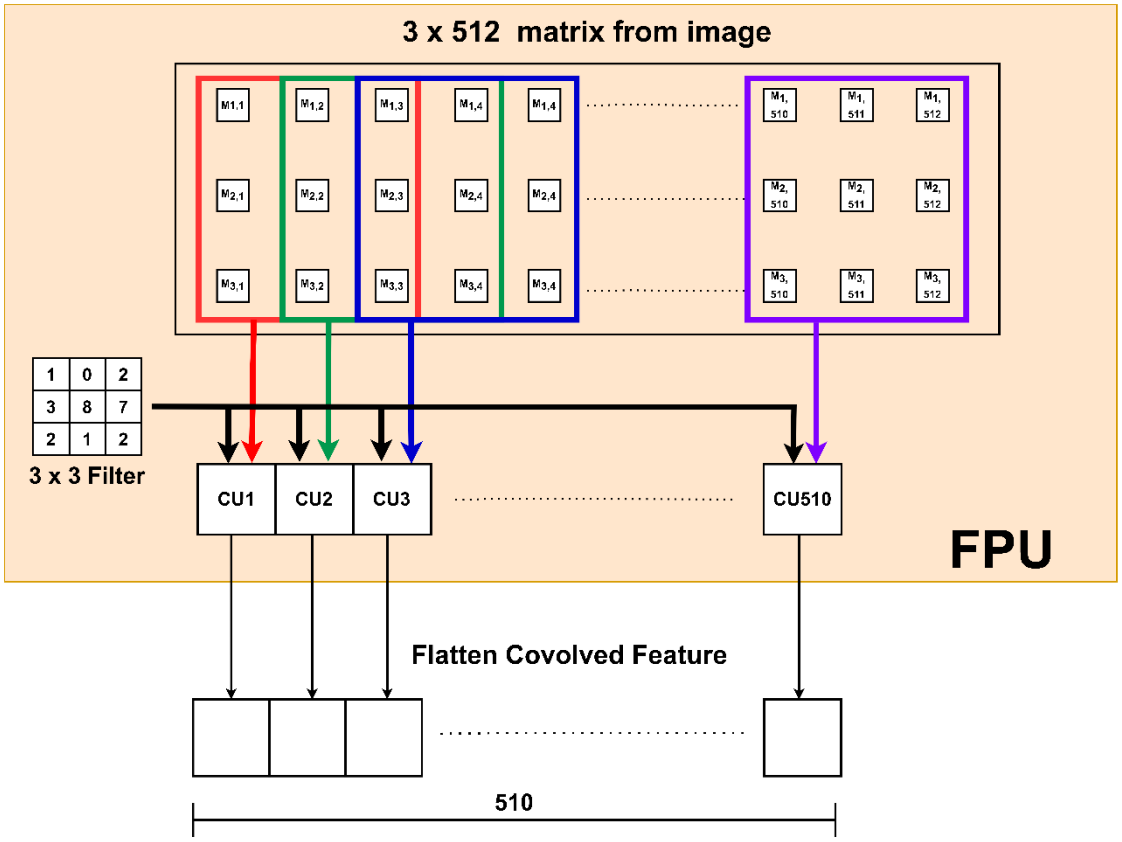

Figure 4. FPU 內部架構，(以512*512的 image 和3*3的 filter 為例)，FPU 當中共有 $512-3+1=510$個 CU，每個 CU 負責在1個 CU cycle 內完成矩陣 Element-wise Multiplication 並將乘積總和輸出成 Flatten Convolved Feature 的其中一值

我們在 U-Net 每個需要運算卷積的層級中設置一個對應大小的 FPU，例：在處理 512*512的 Image 以及3*3的 filter 卷積的層級中設置一個含有 $512-3+1=510$個 CU

的 FPU，其中每個 CU 負責處理3*3矩陣的 Element-wise Multiplication，由於每層僅有一個 FPU，所以總共需要$(N-f+1)$個 CU cycle 來完成一張$N*N$的 Image 的對$f*f$的 Filter 的卷積。

3.2 Pipelining

在前述所談論的設計下，對 U-Net 中的每個需要處理卷積的層級來說，假設該層須處理的 Image 寬度為 N，Filter 寬度為 f，總共需要$(N-f+1)$個 CU cycle 來完成該層級的卷積，但值得注意的是我們可以提前先把已經完成 Flatten convolved feature 送到下一層級，如此一來，當下一層級收集到了足夠多來自上層的 flatten convolved feature 後，便可以啟動該層級的 FPU 來開始進行卷積運算，不必等到上一層級的卷積完全跑完，省下等待的時間，實現 Pipelining 加速。

4. Results

回歸到本研究的目的 — 加速 U-Net 的卷積運算時間，所以我們以 clock cycle 為比較單位，計算並比較我們的設計與 CPU 加 C++的配置花在 U-Net 卷積上的 clock cycle 數，來證明我們的設計有達到加速效果。

我們先以簡易的雙層 U-Net 架構來計算 clock cycle 數並比較，並在後面討論層數往上提升的話，CPU 加 C++配置與我們的設計所需的 clock cycle 數分別是如何變化。

假設第一層要進行卷積的 image 大小為$N*N$，filter 大小為$f1*f1$，第二層的 filter 大小為$f2*f2$，在 CPU 加 C++配置中，每次滑動 filter 需要1個 clock cycle，也就是說要使 filter 滑過整個 image 所需的時間為$(N-f1+1)^2$ 個 clock cycle，並且需要等到第一層的卷積全部算完後第二層才會開始計算，所以對 CPU 加 C++配置而言，完成兩層卷積所需的時間為: $(N-f1+1)^2 + ([N-f1+1]-f2+1)^2$個 clock cycle。

對我們的設計而言，首先 FPU 的運用可以使第一層卷積運算所需的 clock cycle 數降到$(N-f1+1)$個 CU cycle，(注: CU cycle 為 CU 做一次運算所花的 clock cycle 數，通常為1)，同時由於 Pipelining 的運用，第二層的卷積不必等到第一層算完再開始，所以第二層的第一個 flatten convolved feature 在第一層算出 f2個 flatten convolved feature(at time = f2 * CU cycle)時就可開始計算，再花一個 CU cycle 即可計算出來(at time = $(f2+1) * CU$ cycle)，同理，第二層最後一個 flatten convolved feature 也只需要等到第一層最後一個 flatten convolved feature 後再加一個 CU cycle 就可以計算出來

(at time = $([N - f1 + 1] + 1) * CU\ cycle$)，(如下 Figure 5. 所示)。

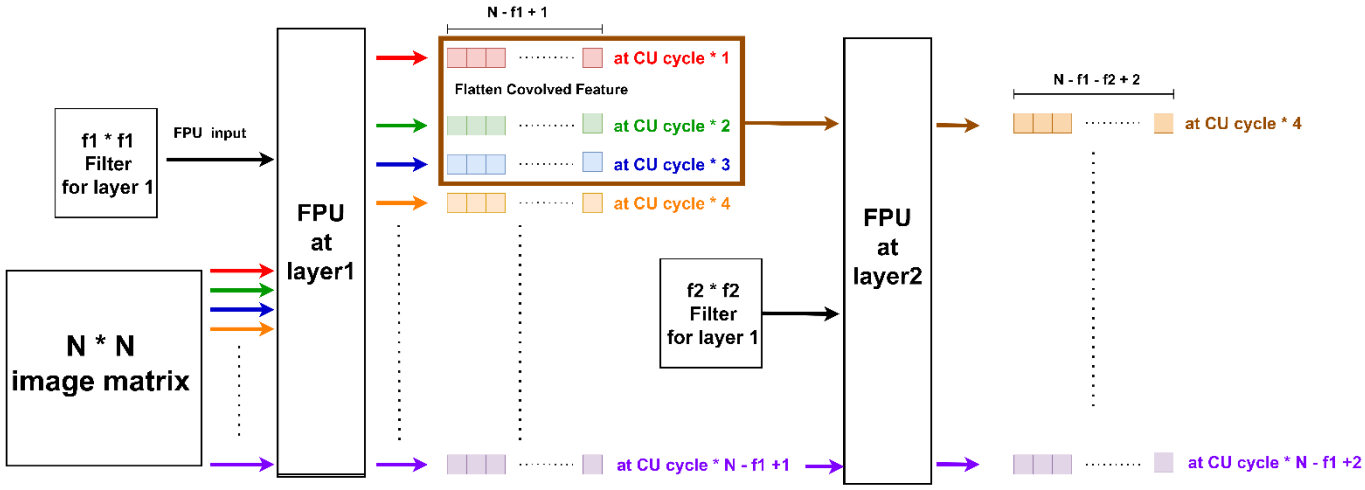

Figure 5. 雙層 U-Net 透過我們的設計在卷積過程中，每個 flatten convolved feature 完成的時間點(以 clock cycle 為單位)

比較兩者的結果，CPU 加 C++所花 clock cycle 數為: $(N - f1 + 1)^2 + ([N - f1 + 1] - f2 + 1)^2$，我們的設計所花 clock cycle 數為: $(N - f1 + 2)$ (假設 CU cycle = 1 clock cycle)，後者所花 clock cycle 數明顯少於前者。

若增加 U-Net 的層數至3層，CPU plus C++完成全部卷積所需要的 clock cycle 數會變為$(N - f1 + 1)^2 + ([N - f1 + 1] - f2 + 1)^2 + ([N - f1 - f2 + 2\ ] - f3 + 1)^2$個 clock cycle，比起雙層增加了$N^2$數量級的 clock cycle，另一方面，我們的設計完成三層卷積所需的時間為$(N - f1 + 3)$個 CU cycle，比起雙層增加了1個 CU cycle，這表示若持續增加 U-Net 的層數，CPU 加 C++會不斷增加N^2數量級的 clock cycle，但我們的設計只會每次加上一個 CU cycle 而已，可想而知對層數越多的 U-Net 來說，我們的設計所帶來的加速效益會越明顯。

因此我們可以證明，相比於 CPU 加 C++配置，我們的設置能確實花更少的 clock cycle 完成卷積計算，並且對於越多層的 U-Net 結構來說，加速效果越強。

5. Conclusion

在這1年的研究中，我們拆解並了解 Stable Diffusion 在執行文字生成圖片任務時各區塊的工作內容以及分析其所花費的時間佔比，找出要加速 Stable Diffusion 最關鍵的區塊 — U-Net，並成功以硬體設計的方式設計出能加速 U-Net 的卷積運算的架構，並透過計算驗證我們的設計比其傳統 CPU plus C++配置能花更少的 clock cycle 去完成 U-Net 的卷積運算，成功達到加速之效果。

6. Review and reflections

在這一年的專題研究中，我們從零到有地發想出一個研究方向、找到研究的目的以及該解決的問題、並在最後實作出能解決問題的研究，這個過程雖然十分艱辛卻讓我們受益良多。

首先，在廣泛閱讀專題題目相關論文時，我們吸收了大量 AI 加速相關的知識，除了擴充我們的見聞外更可能對未來的研究有所幫助，此外，透過大量閱讀論文，我們也可以一定程度地去洞悉論文撰寫者是如何看待 AI 加速相關課題、如何發現問題所在並且設計出能解決問題的方法，能夠幫助我們在未來研究上能夠更快地切入重點，找出問題並設法解決，在報告已論文摘要給指導教授時，也是一個訓練我們表達與組織能力的大好機會，並且教授也會對我們的報告內容做出指點並提出建議，以此磨練我們抓重點及閱讀論文的能力。

在實作階段時，我們能夠利用自己所學及專業能力去設計出解決問題的方法，並實際將其實現，這歸功於我們在專題研究期間，一有想法就會去驗證和試錯，如此在這一年間培養出了一定的實作經驗和能力，此外，組員間的分工以及互相配合更是難得的合作經驗，能使我們未來在團體 project 中能有更出色的團隊能力。

總體來說，這一年來的專題研究讓我們得到許多寶貴的經驗及磨練，我們期望我們能帶著這一年來的所學與體悟，讓我們在未來往學術的探求路途上，能夠更有能力及自信地面對接下來的挑戰和課題。

7. Reference

[1] Rombach; Blattmann; Lorenz; Esser; Ommer (June 2022). High-Resolution Image Synthesis with Latent Diffusion Models. International Conference on Computer Vision and Pattern Recognition (CVPR). New Orleans, LA. pp. 10684 – 10695

[2] Ronneberger O, Fischer P, Brox T (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation