Universidad de Santiago de Chile (2025), 1–1

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

ARTICLE TYPE

Tarea 1: Convolución de Winograd

Felipe Cubillos Arredondo

Procesamiento de Señales e Imágenes (1/2025)

Abstract

En el presente informe, se hará una comparación en tiempo de ejecución respecto a 3 métodos de computar la convolución entre dos señales discretas en una dimensión, Winograd y Matriz de Toeplitz. Las implementaciones, tiempos tomados y gráficos fueron realizados utilizando Matlab. Finalmente, llegando a que la convolución de Winograd es un método superior a la convolución por multiplicación de matrices.

1. Introducción y Solución Propuesta

La convolución de Winograd reduce operaciones multiplicativas mediante transformaciones algebraicas. Para filtros de largo r, el algoritmo $F(m,r)$ requiere $m+r-1$ multiplicaciones. En este trabajo, se implementó $F(2,3)$, que calcula 2 salidas usando 4 multiplicaciones, dividiendo el filtro en segmentos de 3 coeficientes. Cömo motivación, en el paper de Lavin y Gray, se utiliza en el contexto de las redes neurona

- $y_1 = m_1 + m_2 + m_3$
- $y_2 = m_2 - m_3 - m_4$

- $m_1 = (x_0 - x_2)h_0$
- $m_2 = (x_1 + x_2)\frac{h_0+h_1+h_2}{2}$
- $m_3 = (x_2 - x_1)\frac{h_0-h_1+h_2}{2}$
- $m_4 = (x_1 - x_3)h_2$

Las matrices de transformación para $F(2,3)$ son:

1. $B^T \in \mathbb{R}^{4\times 4}$ (señal de input),
2. $G^T \in \mathbb{R}^{4\times 3}$ (filtro),
3. $A^T \in \mathbb{R}^{2\times 4}$ (inversa)

Estas dimensiones permiten procesar ventanas de 4 muestras para generar 2 salidas.

Con las ecuaciones desglosadas, podemos generalizar por medio de zero-padding correspondiente y corriendo la señal dependiendo del largo del filtro y así obtener $y[n]$ **El filtro debe ser múltiplo de 3 porque el algoritmo $F(2,3)$ opera sobre segmentos de 3 coeficientes.** Si el largo no es divisible por 3, la descomposición en bloques falla, como se verifica en el código con `mod(M,3)  = 0`. Para calcular las siguientes dos muestras $(y3, y4)$, se emplea "sliding windows", la ventana se desplaza dos posiciones $(i \to i+2)$, y se repiten los pasos previos.

1. Generar la ventana de entrada extrayendo un bloque de 4 muestras consecutivas de la señal de entrada x, con padding correspondiente.
2. Por medio de las matrices B y T, se obtienen (m_1, m_2, m_3, m_4).
3. Por medio de las ecuaciones iniciales, obtener y_1, y_2.
4. Finalmente, se corre la ventana en 2 posiciones y se repite.

Esto permite procesar la señal completa mediante el método overlap-add, acumulando resultados solapados.

2. Experimentos Realizados y Tiempos Logrados

Llevando a la práctica la teoría revisada se implementaron las siguientes funciones:

- `toep_convolucion(x,h)`: Encargado de computar la convolución por medio de $H^T x = y[n]$.
- `winograd_ventana(x, h)`: Encargado de computar los primeros 6 valores de la convolución, solo es aplicable a filtros de largo 3.
- `winograd(x, h)`: Aplicando `winograd_ventana` se puede ir deslizando los índices en subfiltros.

Menos sustancialmente, se deja la opción al usuario entre introducir las señales o generar una señal x de largo 8 o un filtro h de largo 6 aleatorio, ambos con valores que oscilan entre -1 y 1.

Mas aún, se implementa un gráfico con una señal de impulso de largo 6 y una señal aleatoria que va del 2^i con $i \in N$, donde $3 \leq i \leq 13$.

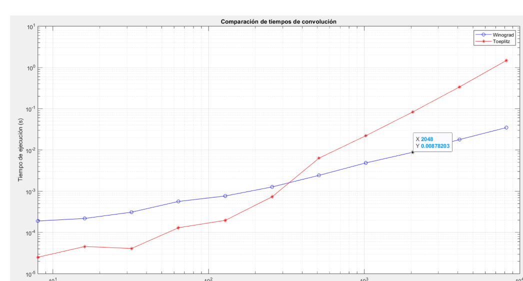

Figure 1. Comparación de tiempos de ejecución (escala logarítmica).

3. Conclusiones

Evidenciado por el gráfico, podemos concluir vemos que el acercamiento algorítmico de Winograd para computar la convolución para señales discretas es **más rápido para muestras de mayor tamaño que con la convolución basada en matriz de Toeplitz.** Justamente por la precomputación de valores y la cantidad reducida de multiplicaciones. En el contexto de convnets presentado en el paper, esto significa una **mejora significativa en los tiempos de ejecución.**