Implementation of FFT on Image Processing and Convolution Neural Network (Report of Final Project MA5104)

Bo-Cyuan Lin (College Students of NCU Mathematics)

Preliminary

0.1 Basic Conception of the Fourier Series

0.1.1 Orthonormal Family

Definition. (Normalized)
$(\nu, <,>)$m inner product space. $\varphi \in \nu$ is called normalized, if $||\varphi|| = \sqrt{<\varphi, \varphi>} = 1$.

Definition. (Orthonormal Family)
$(\nu, <,>)$, inner product space, $\varphi_0, \varphi_1, \varphi_2, ... \in \nu$ is called an orthonormal family, if it satisfies:

$$\begin{cases} <\varphi_i, \varphi_j> = 1, \ if \ i = j \\ <\varphi_i, \varphi_j> = 0, \ if \ i \neq j \end{cases}$$

0.1.2 Definition of Fourier Series and Fourier Coefficients

Definition. (Complete)
An orthomormal family $\varphi_0, \varphi_1, \varphi_2, ...$ in an inner product space ν is complete, if $\forall \ f \in \nu, \ f = \sum_{k=0}^{\infty} c_k \varphi_k$.

Definition. (Fourier Series and Fourier Coefficient)
We call $\sum_{k=0}^{\infty} < f, \varphi_k > \varphi_k$ Fourier Series of f with respect to $\varphi_0, \varphi_1, \varphi_2, ...$ and $< f, \varphi_k >$ is the Fourier coefficients.

Note.

1. $\{\varphi_k\}$ is complete $\Leftrightarrow \forall \ f \in \nu, \ ||f - \sum_{k=0}^{n} < f, \varphi_k > \varphi_k|| \to 0, \ as \ n \to \infty$.
2. $\nu = L^2, \ f : [a, b] \to \mathbb{C}$ with $< f, g >= \int_a^b f(x)\overline{g(x)}dx$ "Fourier Series" is write each $f \in \nu$ into $f = \sum_{k=0}^{\infty} c_k \varphi_k$, where $c_k \in \mathbb{C}$ and $\varphi_0, \varphi_1, \varphi_2, ...$ is a given orthonormal family.
3. If $\forall \ f \in \nu, \ f = \sum_{k=0}^{\infty} c_k \varphi_k$, then $\varphi_0, \varphi_1, \varphi_2, ...$ is complete.
4. First job is to determine the "Fourier Coefficients" c_k in $f = \sum_{k=0}^{\infty} c_k \varphi_k$.

0.1.3 Exponential system and Trigonometric system

The following two systems are closed related, the trigonometric system can be obtained by taking the real part and imaginary part of the exponential system.
Let $f \in L^2, \ f : [-\frac{T}{2}, \frac{T}{2}](or \ [0, T]) \to \mathbb{C}$ with period T, then

1. Exponential system :
   $F_{exp}(t) = \sum_{k=-\infty}^{\infty} c_k e^{\frac{2\pi i k t}{T}}$, where $c_k = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) e^{\frac{-2\pi i k t}{T}} dt$.

2. Trigonometric system :
$F_{tri}(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k cos(\frac{2\pi ikt}{T}) + b_k sin(\frac{2\pi ikt}{T}))$, where

$$\begin{cases} a_k = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)cos(\frac{2\pi ikt}{T})dt,\ k \in [0,\infty) \\ b_k = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)sin(\frac{2\pi ikt}{T})dt,\ k \in [1,\infty) \end{cases}$$

Note. In (Marsden) Elementary Classical Analysis Theorem 10.3.1 (Mean Completeness Theorem) proves that these two systems are complete.

0.2 Simple Conception of Fourier Transform (FT)

Definition. (Exponential Fourier Series)
$F(t) = \sum_{k=-\infty}^{\infty} c_k e^{\frac{2\pi ikt}{T}}$, where $c_k = \frac{1}{T}\int_{\frac{T}{2}}^{-\frac{T}{2}} f(t)e^{\frac{-2\pi ikt}{T}}dt,\ k \in \mathbb{Z},\ t \in [-\frac{T}{2}, \frac{T}{2}],\ f \in L^2$, satifies Dirichlet Lemma (i.e. $||f - F||^2 = 0$).

0.2.1 Relation between the Fourier Series and Fourier Transform

Generally speaking, since f in the series whose integral is bounded, the Fourier Transform is consider the period T attains infinity.

Derivation. (Derivation of the Fourier Transform)
Let $f_T(t)$ be the $T - period$ function, $t \in [-\frac{T}{2}, \frac{T}{2}],\ f_t(t) = f(t)$.
Define $\nu_k = \frac{k}{T}\ s.t.\ \nabla\nu = \nu_{k+1} - \nu_k = \frac{1}{T}$. Then the Fourier Series of $f_T(t)$:
$F(t) = \sum_{k=-\infty}^{\infty} c_k(T)e^{\frac{2\pi ikt}{T}}$, where $c_k(T) = \frac{1}{T}\int_{\frac{T}{2}}^{-\frac{T}{2}} f(t)e^{\frac{-2\pi ikt}{T}}dt,\ k \in \mathbb{Z}$.

$$\begin{aligned} \Rightarrow F(t) &= \sum_{k=-\infty}^{\infty} \frac{1}{T}\int_{\frac{T}{2}}^{-\frac{T}{2}} f(t)e^{\frac{-2\pi ikt}{T}}dt e^{\frac{2\pi ikt}{T}} \\ &= \sum_{k=-\infty}^{\infty} \int_{\frac{T}{2}}^{-\frac{T}{2}} f(t)e^{\frac{-2\pi ikt}{T}}dt\ e^{\frac{2\pi ikt}{T}}\frac{1}{T} \\ &= \sum_{k=-\infty}^{\infty} \int_{\frac{T}{2}}^{-\frac{T}{2}} f(t)e^{-2\pi i\nu_k t}dt\ e^{2\pi i\nu_k t}\nabla\nu \end{aligned}$$

As $T \to \infty \Rightarrow \nabla\nu \to d\nu$ with $\sum_{k=-\infty}^{\infty} \to \int_{-\infty}^{\infty}$

$$\Rightarrow f(t) = F(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t)e^{-2\pi i\nu t}dt\ e^{2\pi i\nu t}d\nu.$$

Note that $\hat{f}(\nu) = \int_{-\infty}^{\infty} f(t)e^{-2\pi i\nu t}dt$, where ν is the frequency and $f(t) = \int_{-\infty}^{\infty}\hat{f}(\nu)e^{2\pi i\nu t}d\nu$.

0.2.2 Definition of the Fourier Transform and Inverse Fourier Transform

Definition. (Fourier Transform)
Let f be a period function and bounded Riemann integrable, then
Fourier Transform:
$\hat{f}(\nu) = \int_{-\infty}^{\infty} f(t)e^{-2\pi i\nu t}dt$, where ν is the frequency.

Definition. (Inverse Fourier Transform)
Let f be a period function and bounded Riemann integrable and $\hat{f}$ be the Fourier Transform of f, then Inverse Fourier Transform:
$f(t) = \int_{-\infty}^{\infty} \hat{f}(\nu) e^{2\pi i \nu t} d\nu$.

Since the computer can only compute the finite dimensional vectors, so we have to turn the Fourier Transform into the Discrete Fourier Transform in next section.

0.3 Discrete Fourier Series (DFT)

0.3.1 Definition of Discrete Fourier Transform

Let the f be the $T-period$ function and bounded Riemann integrable over $[0, T]$, since f is period, we can let $x_j = f(t_j),\ where\ t_j = \frac{jT}{N},\ j = 1, 2, ..., N-1$ and normalize the T s.t. $T = 1$ (i.e. $t_j = \frac{j}{N}$).

Definition. (Discrete Fourier Transform)
Let a sequence of N complex numbers $\{x_0, x_1, ..., x_{N-1}\}$, and sequence $\{y_k\}$ define by: $y_k = \sum_{j=0}^{N-1} x_j e^{\frac{-2\pi i k j}{N}}$.

Consider the relation between x_j and y_k is the invertible linear transform, so we can use $N \times N$ invertible matrix to express the Fourier Transform.

Definition. (Fourier matrix)
Use the Euler Formula we have $\omega = \omega(N) = e^{\frac{-2\pi i}{N}}$, and express the DFT with the form $y = Fx$ as the following:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \end{pmatrix}_{N\times 1} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{(N-1)} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & & & & \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)^2} \end{pmatrix}_{N\times N} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{pmatrix}_{N\times 1}$$

Where F is a symmetric complex matrix.

0.3.2 Discrete Inverse Fourier Transform

Derivation. (Derivation of the Discrete Inverse Fourier Transform)
Let $\{y_k\}_{k=0}^{N-1}$ be the DFT of the data sequence $\{x_j\}_{j=0}^{N-1}$. To show:

$$x_j = \frac{1}{N} \sum_{k=0}^{N-1} y_k e^{\frac{2\pi i k j}{N}}$$

Consider $\sum_{k=0}^{N-1} y_k e^{\frac{2\pi i k j}{N}} = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} x_i e^{-\frac{2\pi i k l}{N}} e^{\frac{2\pi i k j}{N}}$

$$= \sum_{l=0}^{N-1} (x_l \sum_{k=0}^{N-1} e^{\frac{2\pi i k (j-l)}{N}}) = N x_j + \sum_{l=0, l\neq j}^{N-1} (x_l \sum_{k=0}^{N-1} e^{\frac{2\pi i k (j-l)}{N}})$$

$$= N x_j + \sum_{l=0, l\neq j}^{N-1} (x_l \frac{e^{2\pi i k (j-l)} - 1}{e^{\frac{2\pi i k (j-l)}{N}} - 1}) = N x_j + 0 = N x_j$$

$$\Rightarrow x_j = \frac{1}{N} \sum_{k=0}^{N-1} y_k e^{\frac{2\pi i k j}{N}}$$

Definition. (Discrete Inverse Fourier Transform)
Let $\{y_k\}_{k=0}^{N-1}$ be the DFT sequence of the $\{x_j\}_{j=0}^{N-1}$, then the DIFT is the following:

$$x_j = \frac{1}{N}\sum_{k=0}^{N-1} y_k e^{\frac{2\pi i k j}{N}}.$$

Definition. (Inverse Fourier matrix)
Use the Euler Formula we have $\omega = \omega(N) = e^{\frac{-2\pi i}{N}}$, and express the DFT with the form $F^{-1}y = x$ as the following:

$$F^{-1} = \frac{1}{N}\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \cdots & \omega^{-(N-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \cdots & \omega^{-2(N-1)} \\ \vdots & & & & \\ 1 & \omega^{-(N-1)} & \omega^{-2(N-1)} & \cdots & \omega^{-(N-1)^2} \end{pmatrix}_{N\times N}$$

Where F is a symmetric complex matrix.

0.3.3 Example of DFT

Example 0.1. Find the DFT of the sequence $\{x_0, x_1\}$ and show that DIFT(DFT($\{x_0, x_1\}$)) = $\{x_0, x_1\}$.
sol:

$$\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}_{2\times 1} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}_{2\times 2}\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}_{2\times 1} = \begin{pmatrix} 1 & 1 \\ 1 & \omega \end{pmatrix}_{2\times 2}\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}_{2\times 1}$$

So, DFT($\{x_0, x_1\}$) = $\{x_0 + x_1, x_0 - x_1\}$.

$$\frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & \omega^{-1} \end{pmatrix}_{2\times 2}\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}_{2\times 1} = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}_{2\times 2}\begin{pmatrix} x_0 + x_1 \\ x_0 - x_1 \end{pmatrix}_{2\times 1} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}_{2\times 1}$$

Hence, DIFT(DFT($\{x_0, x_1\}$)) = DIFT($\{x_0 + x_1, x_0 - x_1\}$) = $\{x_0, x_1\}$.

Note. (Inverse Discrete Fourier Transform Processing)
Compute $\overline{y} \to x = \frac{1}{n}\overline{DFT(\overline{y})}$, where x is the original data and y is the DFT of x.

1 Fast Fourier Transform (FFT)

If we compute the DFT directly with computer, it may cost much of time. The computational complexity of general DFT is $O(N^2)$.

1.1 Cooley-Tukey Algorithm

The main idea about the Cooley-Tukey Algorithm is splitting the DFT into two sub-DFT with both lengths equal $\frac{N}{2}$. Before we do that, we need some properties about the complex roots.

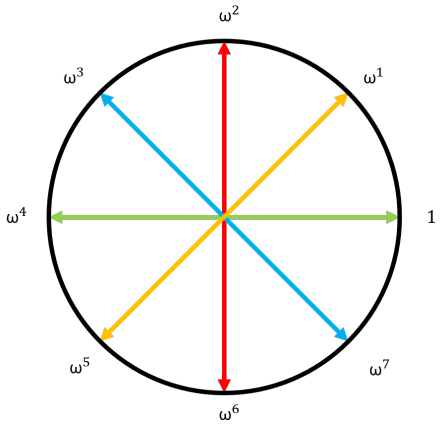
Note. (Some Properties about complex roots)

1. Let the complex set $\{1, \omega, \omega^2, ..., \omega^{N-1}\}$ be the roots of $z^N = 1$, actually they are lie on the unit circle in the complex plane. Moreover, $\{1, \omega^2, \omega^4, ..., \omega^{N-2}\}$ is the roots of $z^{\frac{N}{2}}$. Hence, based on this conception we have $m = 2^k,\ k \in \mathbb{Z}^+,\ \{1, \omega^m, \omega^{2m}, ..., \omega^{N-m}\}$ is a root of $z^{\frac{N}{m}}$.

2. Since the ω has period, if $k \geq N$ we have $\omega^k = \omega^{k \ mod \ N}$.
   For example, If $N = 8$, then $\omega^8 = 1, \omega^9 = \omega, ...$

3. ω is symmetric (i.e. $\omega^{k+\frac{N}{2}} = -\omega^k$). Since Euler Formula we have $\omega^p = e^{\frac{2\pi ip}{N}} = cos(\frac{2\pi ip}{N}) + isin(\frac{2\pi ip}{N})$.
   $\Rightarrow \omega^{k+\frac{N}{2}} = e^{2\pi\frac{k+\frac{N}{2}}{N}} = e^{2\pi\frac{k}{N}+\pi} = -e^{2\pi\frac{k}{N}} = -\omega^k$.
   For example, If $n = 8$, then $\omega^4 = -1, \omega^5 = -\omega, \omega^6 = -\omega^2, \omega^7 = -\omega^3$.

Example 1.1. (Cooley-Tukey Algorithm with $N = 8$)
By Property 2 and 3,



$$\Rightarrow F_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^3 & \omega^3 & \omega^2 & \omega \end{pmatrix}_{8\times 8} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & -1 & -\omega & -\omega^2 & -\omega^3 \\ 1 & \omega^2 & -1 & -\omega^2 & 1 & \omega^2 & -1 & -\omega^2 \\ 1 & \omega^3 & -\omega^2 & \omega & -1 & -\omega^3 & \omega^2 & -\omega \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -\omega & \omega^2 & -\omega^3 & -1 & \omega & -\omega^2 & \omega^3 \\ 1 & -\omega^2 & -1 & \omega^2 & 1 & -\omega^2 & -1 & \omega^2 \\ 1 & -\omega^3 & -\omega^2 & -\omega & -1 & \omega^3 & \omega^2 & \omega \end{pmatrix}_{8\times 8}$$

By Property 1,

$$\Rightarrow F_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^6 & \omega^4 & \omega^2 \end{pmatrix}_{4\times 4} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^2 & -1 & -\omega^2 \\ 1 & -1 & 1 & -1 \\ 1 & -\omega^2 & -1 & \omega^2 \end{pmatrix}_{4\times 4}$$

We may found that the 0,2,4,6 columns constructed by F_4. So we rearrange the F_8. Consider the rearrange matrix:

$$\Rightarrow R_8^T = \begin{pmatrix} e_0 & e_2 & e_4 & e_6 & e_1 & e_3 & e_5 & e_7 \end{pmatrix}_{8\times 8} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}_{8\times 8}$$

$$\Rightarrow F_8R_8^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega^2 & -1 & -\omega^2 & \omega & \omega^3 & -\omega & -\omega^3 \\ 1 & -1 & 1 & -1 & \omega^2 & -\omega^2 & \omega^2 & -\omega^2 \\ 1 & -\omega^2 & -1 & \omega^2 & \omega^3 & \omega & -\omega^3 & -\omega \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & \omega^2 & -1 & -\omega^2 & -\omega & -\omega^3 & \omega & \omega^3 \\ 1 & -1 & 1 & -1 & -\omega^2 & \omega^2 & -\omega^2 & \omega^2 \\ 1 & -\omega^2 & -1 & \omega^2 & -\omega^3 & -\omega & \omega^3 & \omega \end{pmatrix}_{8\times 8} = \begin{pmatrix} F_4 & D_4F_4 \\ F_4 & -D_4F_4 \end{pmatrix}_{8\times 8}, (D_4 = diag(1,\omega,\omega^2,\omega^3)_{4\times 4})$$

By Property 1, consider the general cases $m = 2^k,\ k \in \mathbb{Z}^+$, since $R_m^{-1} = R_m^T$.

$$\Rightarrow F_m = \begin{pmatrix} F_{\frac{m}{2}} & D_{\frac{m}{2}}F_{\frac{m}{2}} \\ F_{\frac{m}{2}} & -D_{\frac{m}{2}}F_{\frac{m}{2}} \end{pmatrix}_{m\times m} R_m, where$$

$$R_m^T = \begin{pmatrix} e_0 & e_2 & e_{m-2} & \cdots & e_1 & e_3 & e_{m-1} & \cdots \end{pmatrix}_{m\times m},\ and\ D_{\frac{m}{2}}^T = diag(1,\omega^{\frac{N}{m}},\omega^{\frac{2N}{m}},...,\omega^{\frac{N}{2}-\frac{N}{m}})_{\frac{m}{2}\times\frac{m}{2}}$$

Let $x_8 = [x_0\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7]$ be the complex sequences of data.

1. Rearrange the sequences:
   $P_8x_8 = [x_0\ x_2\ x_4\ x_6\ x_1\ x_3\ x_5\ x_7] = [x_4^{even} \mid x_4^{odd}]$.

2. Divide-and-Conquer:

$$F_8x_8 = \begin{pmatrix} F_4 & D_4F_4 \\ F_4 & -D_4F_4 \end{pmatrix}_{8\times 8} P_8x_8 = \begin{pmatrix} F_4 & D_4F_4 \\ F_4 & -D_4F_4 \end{pmatrix}_{8\times 8} \begin{pmatrix} x_4^{even} \\ x_4^{odd} \end{pmatrix}_{8\times 8} = \begin{pmatrix} F_4x_4^{even} + D_4F_4x_4^{odd} \\ F_4x_4^{even} - D_4F_4x_4^{odd} \end{pmatrix}_{8\times 8}$$

   Use 1 on $F_4x_4^{even}$ *and* $F_4x_4^{odd}$
   $\Rightarrow F_4x_4^{even} = [x_0\ x_4\ x_2\ x_6] = [x_2^{even^{even}} \mid x_2^{even^{odd}}]$ *and* $F_4x_4^{odd} = [x_1\ x_5\ x_3\ x_7] = [x_2^{odd^{even}} \mid x_2^{odd^{odd}}]$.
   Then same with the 2, we will obtain:

$$D_2 = \begin{pmatrix} 1 & 0 \\ 0 & \omega^2 \end{pmatrix}_{2\times 2},\ and\ F_2x_2 = \begin{pmatrix} 1 & 1 \\ 1 & \omega^4 \end{pmatrix}_{2\times 2} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}_{2\times 2}$$

   Both of them are easily to compute, then we will obtain same result with the DFT.

Note. (Radix-2 FFT)
The former called Radix-2 FFT for $N = 2^r$, which divides the DFT into $r = log_2N$ layers, and use $\frac{N}{2}$ multiplies in each layers. So we have the complexity equals $\frac{N}{2}log_2N$.

Remark. (Formula of FFT Radix-2)
Let $\omega_N = e^{-\frac{2\pi i}{N}}$, then both of y_k and $y_{k+\frac{N}{2}}$ has $\frac{N}{2}$ terms.

$$\begin{aligned} y_k &= \textstyle\sum_{n=2t}\omega_N^{kn}x_n + \sum_{n=2t+1}\omega_N^{kn}x_n \\ &= \textstyle\sum_t \omega_{\frac{N}{2}}^{kt}x_{2t} + \omega_N^k\sum_t\omega_{\frac{N}{2}}^{kt}x_{2t+1} \\ &= F_{even}(k) + \omega_N^kF_{odd}(k) \end{aligned}$$

$$y_{k+\frac{N}{2}} = F_{even}(k) - \omega_N^kF_{odd}(k),\ and\ y_k = F_{even}(k) + \omega_N^kF_{odd}(k)$$

Note.
For the $n \times n$ image with $n = 2^p$ (Radix-2), where p is the amounts of the layer about butterfly diagram, we have the complexity as the following:

FT	Complexity of Computation
DFT	n^2
FFT	$\frac{n}{2}log(n)$

Table 1: Computing Complexity of DFT and FFT

1.2 Butterfly Diagram

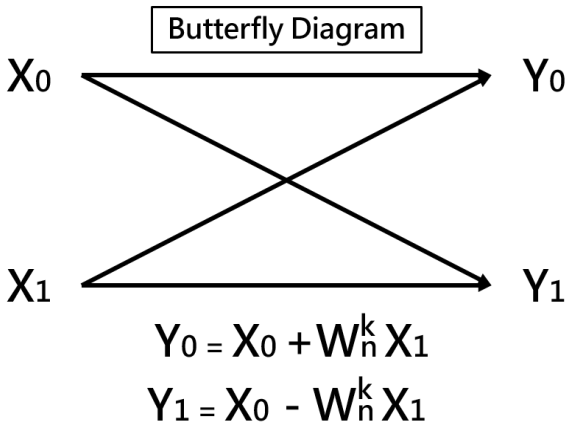
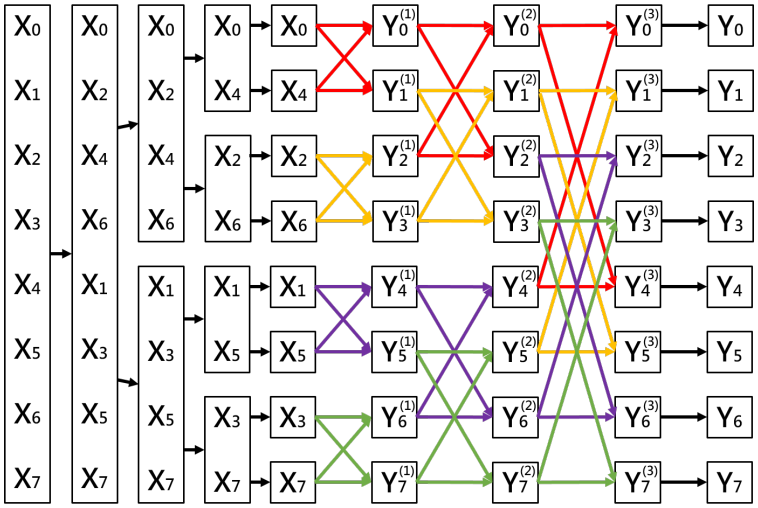


Figure 1: Butterfly Diagram

Butterfly diagram for $N = 8$:



Note. (Possibility of Parallel Butterfly Diagram)
For the different color in the graph above, we can calculate it in parallel.

2 Implementation of FFT on Image Processing

2.1 Ideal Low-Pass

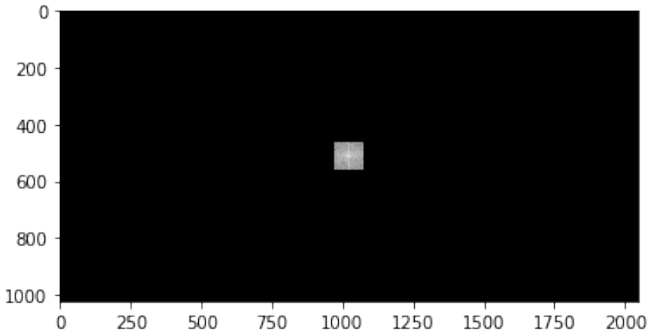


Figure 2: Low-Pass (R = 50)

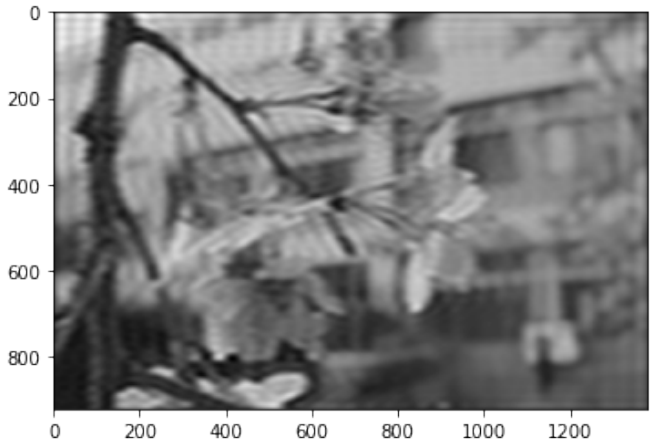


Figure 3: Inverse Low-Pass (R = 50)

2.2 Ideal High-Pass

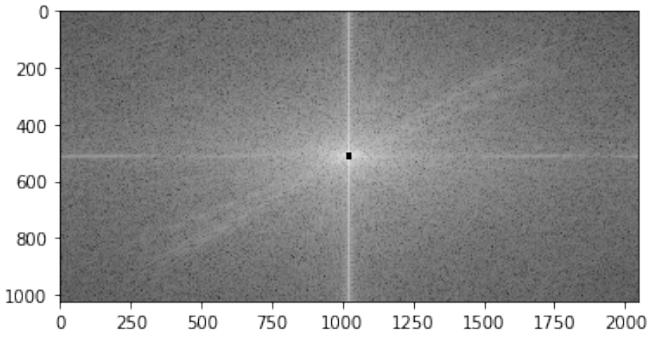


Figure 4: High-Pass (R = 10)

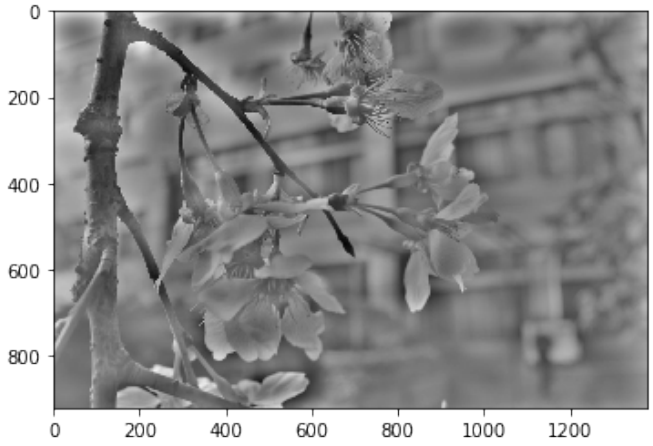


Figure 5: Inverse High-Pass (R = 10)

2.3 Image Compression

We can use the property of frequency data to shift it, then remain the size what we want with low-pass.

Algorithm 1: Image Compression($\times 1/2$)

Input: $n \times n$ 2D-image $X \in \mathbb{R}^{n \times n}$
Output: $n/2 \times n/2$ 2D-image $x \in \mathbb{R}^{n/2 \times n/2}$

1 $\hat{X} \leftarrow FFT2D(x)$
2 $\hat{Y} \leftarrow$ **ShiftSpectrum**(X).
3 $y \leftarrow$ **SelectRange(1/2)**(Y).
4 $\hat{x} \leftarrow$ **ShiftSpectrum**(y).
5 $x \leftarrow IFFT2D(\hat{x})$

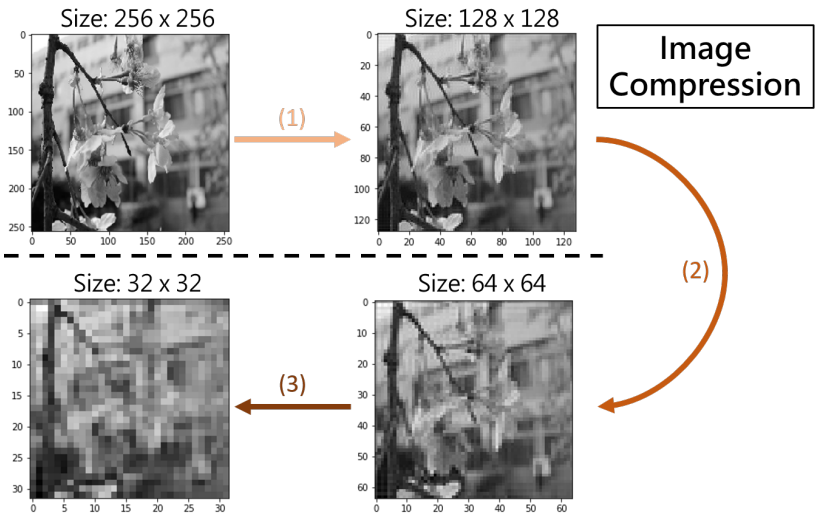


Figure 6: Image Compression through FFT

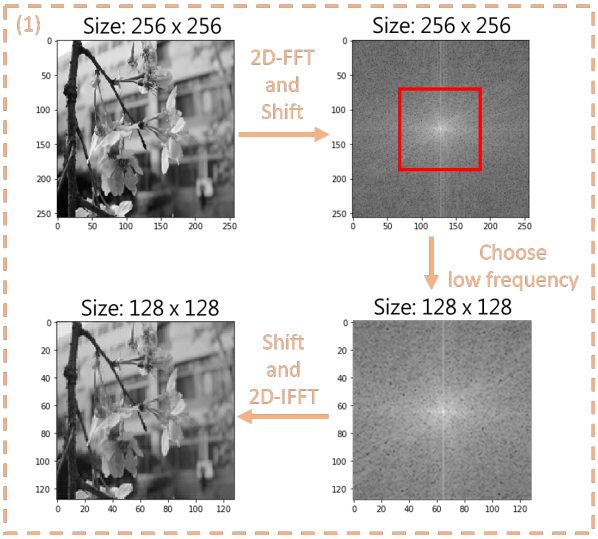


Figure 7: Process (1) of Image Compression through FFT

3 FFT on Convolution Neural Networks

3.1 FFT on Convolution Layer

Theorem 3.1. (Convolution)
Let $f, g \in L^1(\mathbb{R}^n)$ be two function with convolution $f * g$, and $\mathcal{F}$ denotes Fourier Transform, where $\mathcal{F}(f), \mathcal{F}(g)$ are the Fourier Transform of f, g. Then,

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

where $\cdot$ denotes point-wise multiplication.

Note. (Fourier Convolution Layer)
Note that discrete analogue of the Convolution Theorem:

$$\mathcal{F}([kernel] * [data]) = \mathcal{F}([kernel]) \cdot \mathcal{F}([data])$$

where $\mathcal{F}$ denotes the Fourier Transform.

The computation of the DFT for $n \times n$ matrix involves n^2 multiplications and $n(n-1)$ addictions. But this can be improved by Cooley-Tukey FFT Algorithm with $\frac{n}{2} log_2(n)$ multiplications and $nlog_2(n)$ addictions, from $O(n^2)$ to $O(nlog(n))$.

Algorithm 2: FFT Convolution

1 $[kernel]^i_{\mathcal{F}} = \mathcal{F}([kernel]^i),\ i = 1, ..., N_{[kernel]},\ where\ N_{[kernel]}$: kernel amount.
2 $[data]^j_{\mathcal{F}} = \mathcal{F}([data]^j),\ j = 1, ..., N_{[data]},\ where\ N_{[data]}$: data amount.
3 $[output]^{i,j}_{\mathcal{F}} = [kernel]^i_{\mathcal{F}} \odot [data]^j_{\mathcal{F}},\ i = 1, ..., N_{[kernel]},\ j = 1, ..., N_{[data]},\ where \odot : Hadamard\ Product.$
4 $[output]^{i,j} = \mathcal{F}^{-1}([output]^{i,j}_{\mathcal{F}}),\ i = 1, ..., N_{[kernel]},\ j = 1, ..., N_{[data]}.$

The algorithm reduces the number of operations gives an increasing relative speed-up for larger kernels.

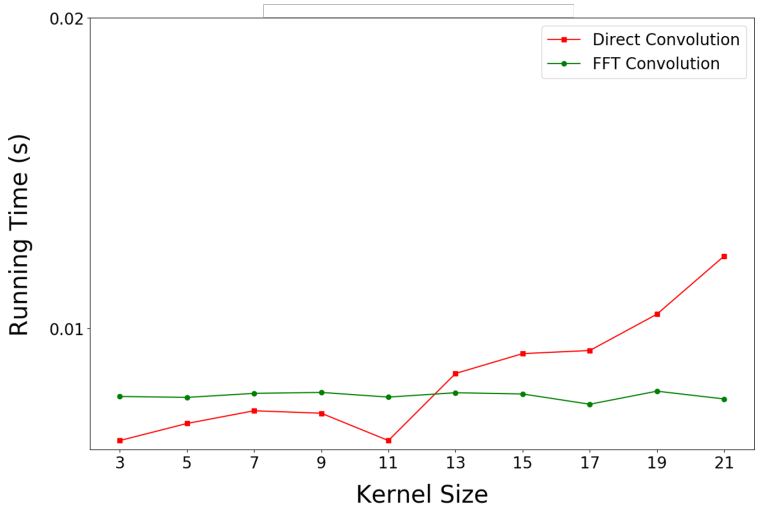


Figure 8: Running Time of Direct and FFT convolution with different kernel size. [Shape of input data:(input size, height, width, channel) = (32,256,256,3)]

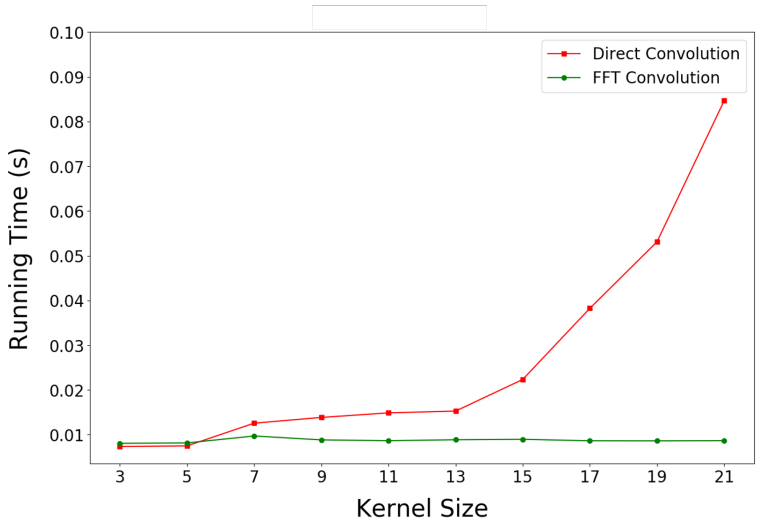


Figure 9: Running Time of Direct and FFT convolution with different kernel size. [Shape of input data:(input size, height, width, channel) = (32,512,512,3)]

3.2 Spectral Pooling Layer

Algorithm 3: Spectral Pooling

Given a complex 3D tensors (x: data set, y: data height, z: data width), $\forall\ image \in x$ we retain the region $[y_{p\ min}, y_{p\ max}] \times [z_{p\ min}, z_{p\ max}]$ as follows:

$$y_{p\ min} = (0.5 - \frac{poolsize}{2}) \times y,\ y_{p\ max} = (0.5 + \frac{poolsize}{2}) \times y$$

$$z_{p\ min} = (0.5 - \frac{poolsize}{2}) \times z,\ z_{p\ max} = (0.5 + \frac{poolsize}{2}) \times z$$

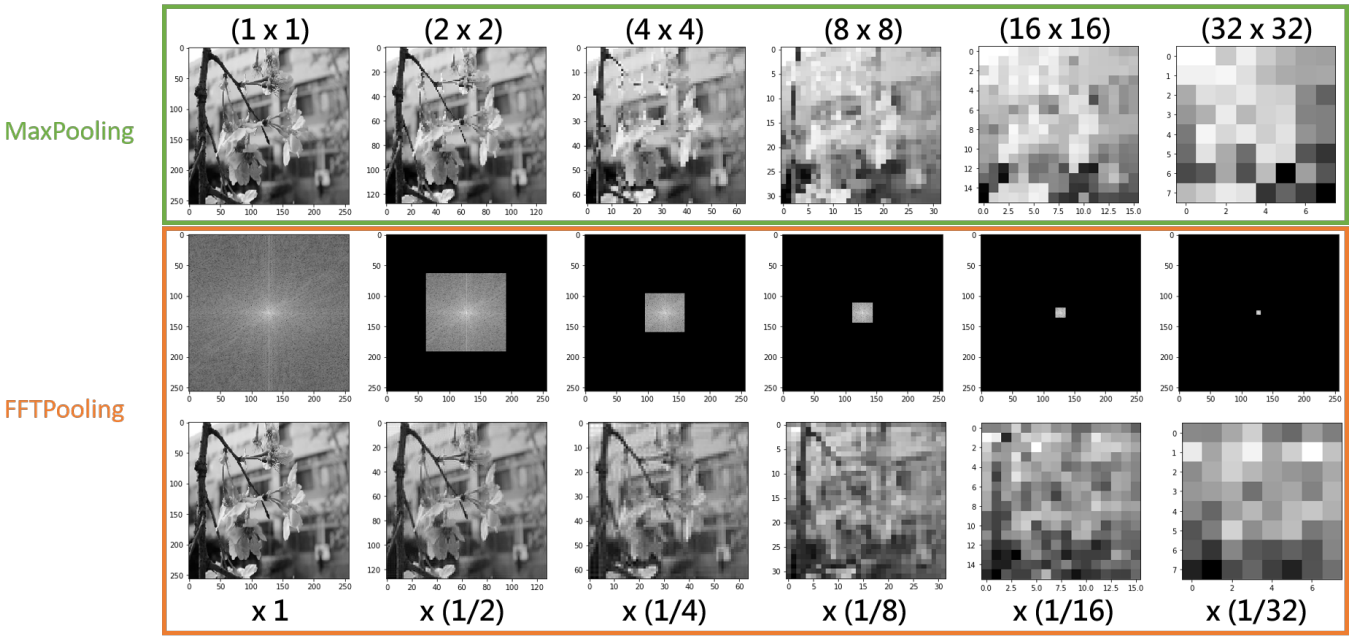
Note.
For the max-pooling, reduce the data size by a quarter by taking the maximum value in the 2×2 matrices. Similarly, we can take $poolsize = 0.25$ s.t. retain the region $[y_{p\ min}, y_{p\ max}] \times [z_{p\ min}, z_{p\ max}]$ as follows:

$$y_{p\ min} = 0.375 \times y,\ \ y_{p\ max} = 0.625 \times y$$

$$z_{p\ min} = 0.375 \times z,\ \ z_{p\ max} = 0.625 \times z$$

which also reduces the data by a quarter.

Comparing with the MaxPooling, Spectral Pooling has better information preservation of the data:



4 Reference

1. Elementary Classical Analysis (2nd Edition); Jerrold E. Marsden, Michael J. Hoffman (1974)

2. M. Mathieu, M. Henaff, and Y. LeCun. Fast training of convolutional networks through ffts. *CoRR(2013)*

3. Oren Rippel, Jasper Snoek and Ryan P. Adams, Spectral Representations for Convolutional Neural Networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), *Advances in neural information processing systems, 28*, pp. 2449–2457

4. Pratt H., Williams B., Coenen F., Zheng Y. (2017) FCNN: Fourier Convolutional Neural Networks. In: Ceci M., Hollmén J., Todorovski bibitL., Vens C., Džeroski S. (eds) *Machine Learning and Knowledge Discovery in Databases*, pp. 786–798

5. http://www.math.ncu.edu.tw/ cchsiao/Course/Fourier_Analysis_051/Fourier_Analysis.pdf
(A concise Lecture Note on Fourier Analysis)

6. https://ccjou.wordpress.com