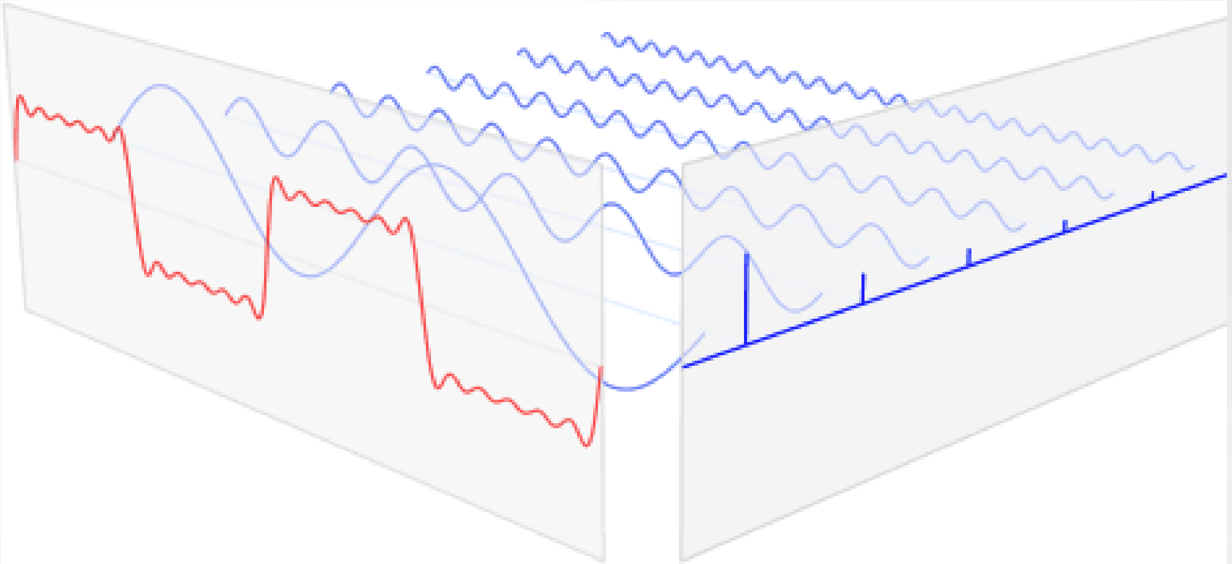
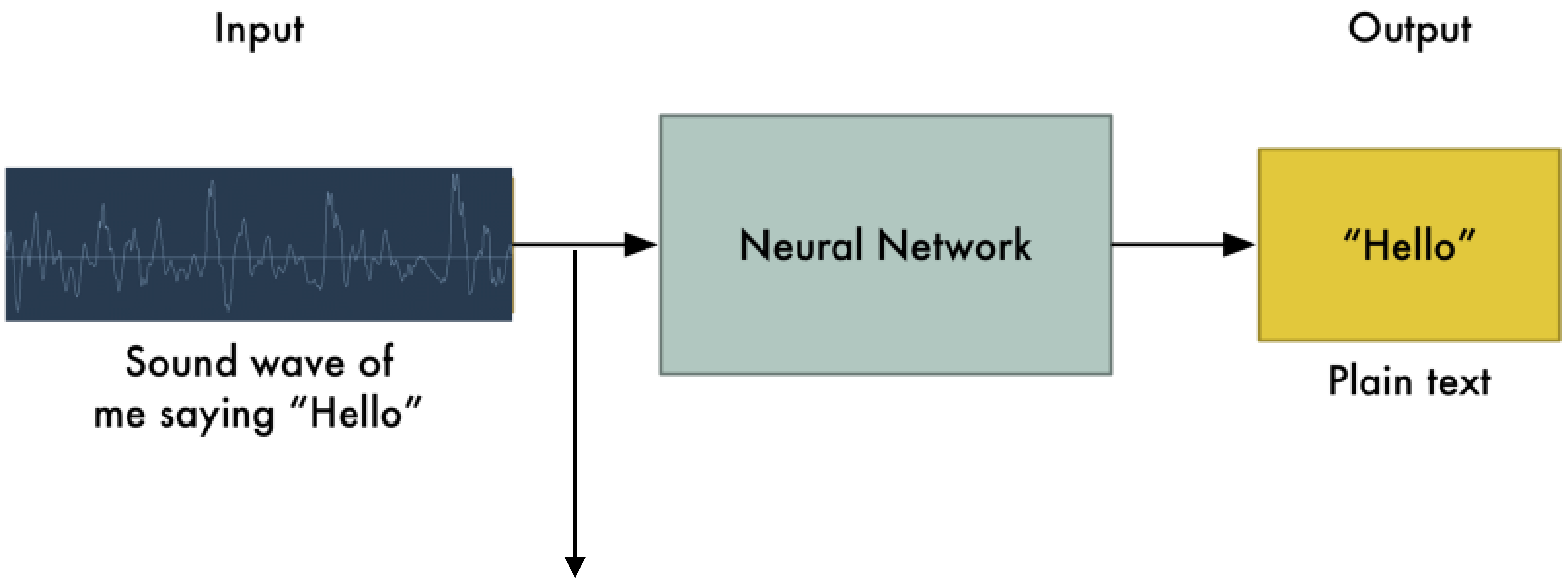
Chapter 09. 시계열을 활용한 딥러닝 (Time Sequence Processing)

FT, FFT, DFT



Copyright FASTCAMPUS Corp. All Rights Reserved

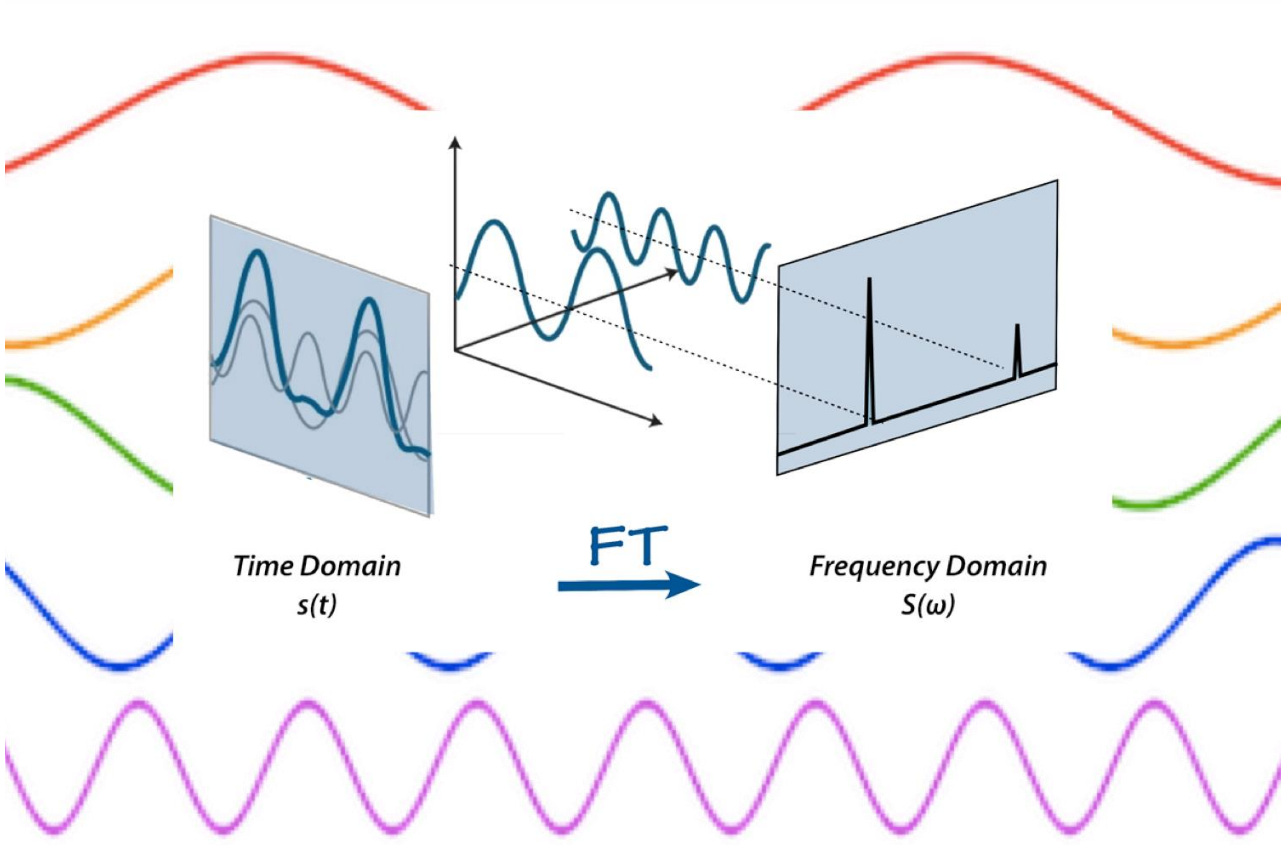
Time Series Deep Learning



Preprocessing

Copyright FASTCAMPUS Corp. All Rights Reserved

Preprocessing



Time Domain

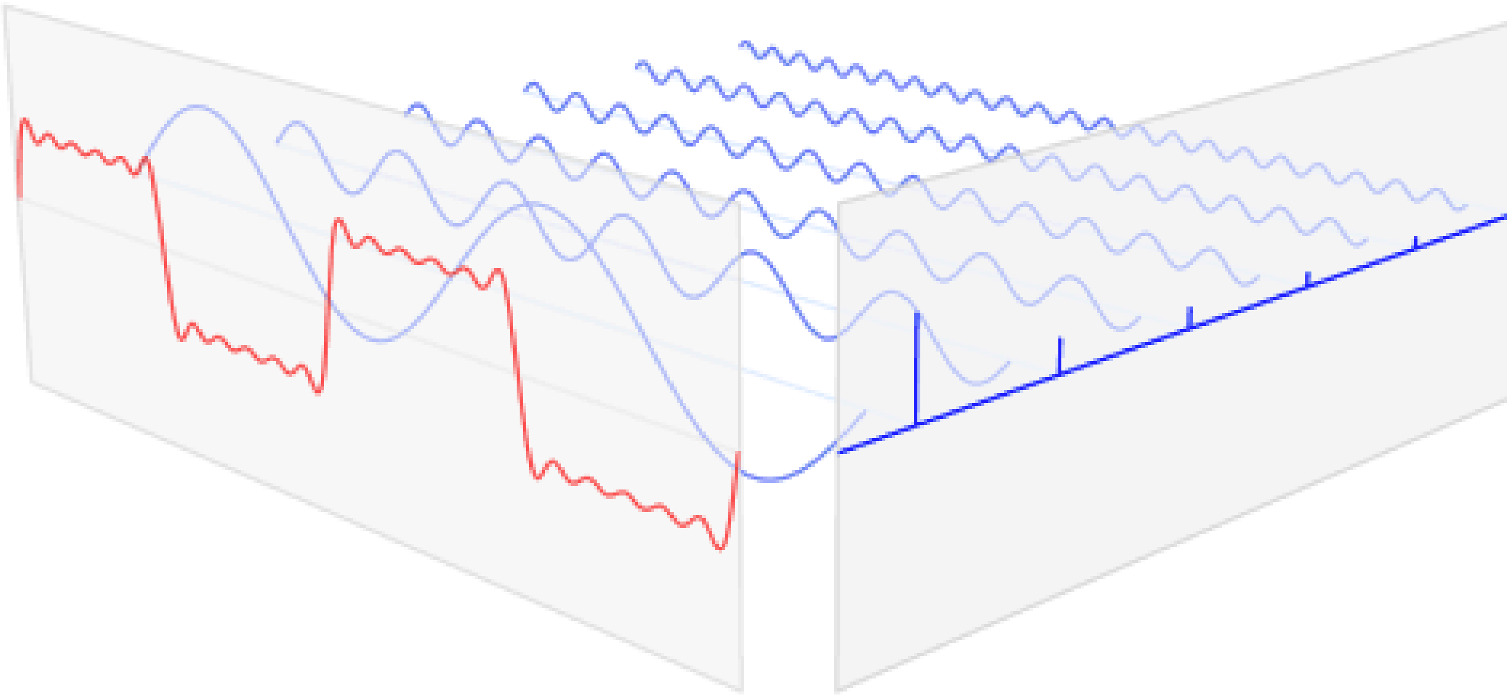
Frequency Domain

Copyright **FASTCAMPUS** Corp. All Rights Reserved

Fourier Transform

https://github.com/mercy-project/korean-audio-sentiment-analysis/blob/eden/Study_Eden_1.md

푸리에 변환은 시간(time) **도메인의 신호를 주파수(frequency) 도메인으로 변환해주는 방법**
임의의 입력 신호를 다양한 주파수를 갖는 주기함수들의 합으로 분해하여 표현



Copyright **FASTCAMPUS** Corp. All Rights Reserved

Fourier Transform

https://darkpgmr.tistory.com/171

https://ghebook.blogspot.com/2012/08/fourier-transform.html

Fourier Series

$$f(t) = \sum_{m=-\infty}^{\infty} F_m e^{im\omega_0 t}$$

Fourier Integral

$$\text{where } F_m = \frac{1}{T}\int_{-T/2}^{T/2} f(t)e^{-im\omega_0 t}\, dt, \ \omega_0 = \frac{2\pi}{T}$$

무한대 주기
(비주기 신호)

$$f(x) = \int_{-\infty}^{\infty} F(u)e^{j2\pi ux}du$$

Inverse Fourier Transform

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}dx$$

Basis Functions

$$e^{j2\pi ux} = \cos 2\pi ux + j \sin 2\pi ux$$

fast campus

Copyright **FASTCAMPUS** Corp. All Rights Reserved

Fourier Transform

신호의 주파수 성분을 알기 위해서 사용

함수 x(t)가 있을 때, 수없이 많은 정현파(코사인과 사인) 성분들이 합쳐진 것이라 생각한다면, 주파수가 f인 정현파(sinusoidal) 성분의 진폭과 위상을 X(f)가 알려 준다.

시스템의 주파수 응답을 알기 위해서

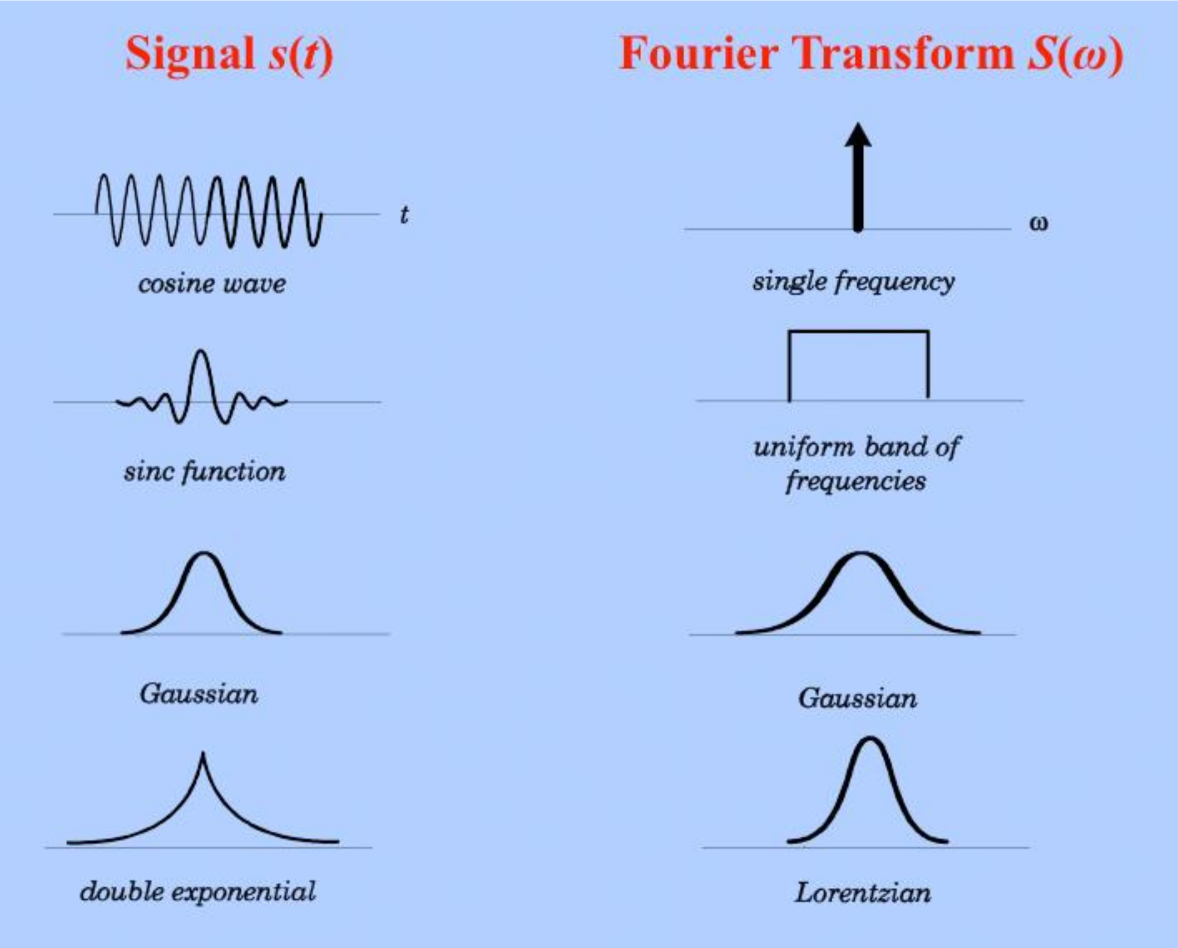
x(t) : 시스템의 '임펄스 응답'(impulse response)
X(f) : 시스템의 '주파수 응답'(frequency response)

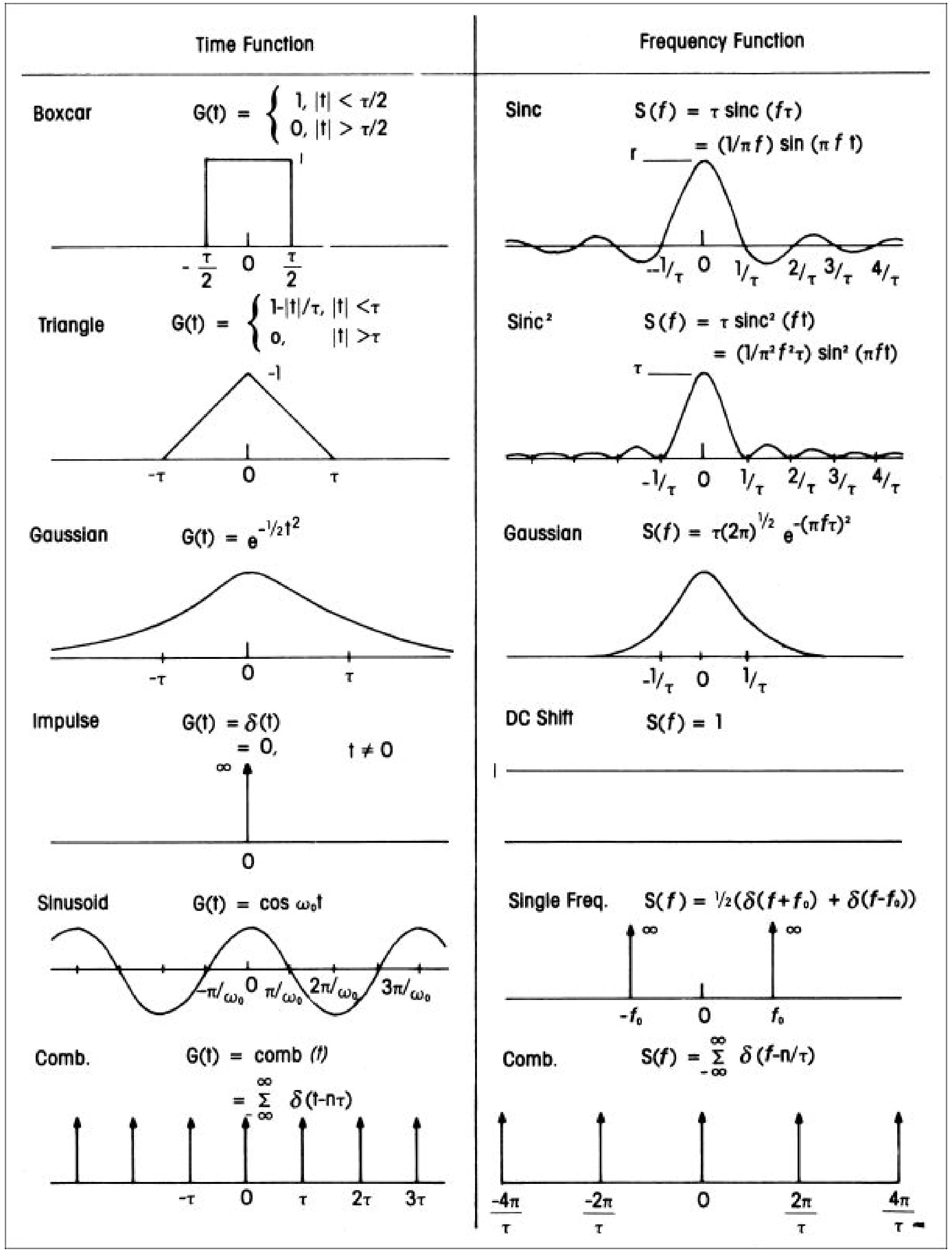
어떤 주파수를 가진 정현파 신호가 시스템에 입력되어 다시 정현파 신호가 출력으로 나올 때에, 입력과 출력 신호의 '진폭의 비율'(이득)과 위상의 차이를 뜻한다.

fast campus

Copyright FASTCAMPUS Corp. All Rights Reserved

Fourier Transform





Copyright **FASTCAMPUS** Corp. All Rights Reserved

Discrete Fourier Transform

DFT(Discrete Fourier Transform) :

DFT 는 유한한 신호 시퀀스의 이산(Discrete) 신호의 푸리에 변환을 구하기 위한 방법

1) 컴퓨터로 푸리에 변환을 할 때 생기는 신호의 길이가 유한하지 않다는 문제점
2) 컴퓨터는 Discrete 한 정보만을 계산할 수 있다는 문제점을 해결한 방법입니다.

fast campus

Copyright **FASTCAMPUS** Corp. All Rights Reserved

Discrete Fourier Transform

1단계 : L개의 sample 로 DTFT(Discrete Time Fourier Transform) 수행

$$X(e^{j\hat{\omega}}) = \sum_{n=0}^{L-1} x[n]\, e^{-j\hat{\omega}n}$$

2단계 : sample 신호의 주파수를 N 개로 나눠보자

$$\widehat{\omega_k} = (2\pi/N)k, k = 0,1,2, \dots , (N-1)$$

3단계 : DFTF 수식에 대입한다.

$$X(e^{j(2\pi/N)k}) = \sum_{n=0}^{L-1} x[n]\, e^{-j(2\pi/N)k}$$

Copyright **FASTCAMPUS** Corp. All Rights Reserved

Fast Fourier Transform

FFT(Fast Fourier Transform)

- DFT의 연산시간이 느려서 고안된 방법.

- sampling된 신호의 전부를 변환시키는 것이 아니라 필요한 신호만을 골라내어서 최소화하여 고속으로 퓨리에 변환을 연산

Ex. 1965년 쿨리-튜키 알고리즘, Rader's FFT algorithm, Bluestein's FFT algorithm

Copyright **FASTCAMPUS** Corp. All Rights Reserved

Fast Fourier Transform

쿨리-튜키 알고리즘(Cooley-Tukey algorithm)

원리에 대한 설명은 다음과 같다.

정의에서 W = e − 2π / N라고 하고 다시 정리하면,

$$f_j = \sum_{k=0}^{n-1} x_k W^{jk} \qquad j = 0, \ldots, n-1$$

예를 들어 N = 4일 때, 이 식을 행렬을

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

지수의 짝홀을 기준으로 위의 식을 다음과 같이 변형할 수 있다.

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^2 & W^1 & W^3 \\ W^0 & W^4 & W^2 & W^6 \\ W^0 & W^6 & W^3 & W^9 \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0W^0 & W^0W^0 \\ W^0 & W^2 & W^1W^0 & W^1W^2 \\ W^0 & W^0 & W^2W^0 & W^2W^0 \\ W^0 & W^2 & W^3W^0 & W^3W^2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ x_1 \\ x_3 \end{bmatrix}$$

fast campus

Copyright FASTCAMPUS Corp. All Rights Reserved

Fast Fourier Transform

쿨리-튜키 알고리즘(Cooley-Tukey algorithm)

이는 다음과 같이 다시 쓸 수 있으므로, N = 4인 DFT는 N = 2인 DFT 두 개를 사용해서 계산할 수 있다.

$$= \begin{bmatrix}1&0&W^0&0\\0&1&0&W^1\\1&0&W^2&0\\0&1&0&W^3\end{bmatrix}\begin{bmatrix}\begin{bmatrix}W_2^0&W_2^0\\W_2^0&W_2^1\end{bmatrix}\begin{bmatrix}x_0\\x_2\end{bmatrix}\\ \begin{bmatrix}W_2^0&W_2^0\\W_2^0&W_2^1\end{bmatrix}\begin{bmatrix}x_1\\x_3\end{bmatrix}\end{bmatrix}$$

이 과정을 재귀적으로 적용하면 N = 2k인 DFT를 O(k, N에 의한) 시간 안에 할 수 있다. 이런 분할 과정을 그림으로 그리면 나비 모양의 그림이 나오기 때문에 나비 연산(Butterfly operation)이라고도 불린다.

Copyright **FASTCAMPUS** Corp. All Rights Reserved

- *Thank you*

Copyright **FASTCAMPUS** Corp. All Rights Reserved