JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 28 No. 4, pp. 1-11, April 2023
https://doi.org/10.9708/jksci.2023.28.04.001

Analyzing performance of time series classification using STFT and time series imaging algorithms

Sung-Kyu Hong*, Sang-Chul Kim**

*Student, Graduate School of Software Technology, Kookmin University, Seoul, Korea
**Professor, School of Computer Science, Kookmin University, Seoul, Korea

[Abstract]

In this paper, instead of using recurrent neural network, we compare a classification performance of time series imaging algorithms using convolution neural network. There are traditional algorithms that imaging time series data (e.g. GAF(Gramian Angular Field), MTF(Markov Transition Field), RP(Recurrence Plot)) in TSC(Time Series Classification) community. Furthermore, we compare STFT(Short Time Fourier Transform) algorithm that can acquire spectrogram that visualize feature of voice data. We experiment CNN's performance by adjusting hyper parameters of imaging algorithms. When evaluate with GunPoint dataset in UCR archive, STFT(Short-Time Fourier transform) has higher accuracy than other algorithms. GAF has 98~99% accuracy either, but there is a disadvantage that size of image is massive.

▸ **Key words**: Time Series Data, GAF, MTF, RP, STFT, Spectrogram, CNN

[요 약]

본 논문은 순환 신경망 대신 합성곱 신경망을 사용하여 시계열 데이터 분류 성능을 분석한다. TSC(Time Series Community)에는 GAF(Gramian Angular Field), MTF(Markov Transition Field), RP(Recurrence Plot)와 같은 전통적인 시계열 데이터 이미지화 알고리즘들이 있다. 실험은 이미지화 알고리즘들에 필요한 하이퍼 파라미터들을 조정하면서 합성곱 신경망의 성능을 평가하는 방식으로 진행된다. UCR 아카이브의 GunPoint 데이터셋을 기준으로 성능을 평가했을 때, 본 논문에서 제안하는 STFT(Short Time Fourier Transform) 알고리즘이 최적화된 하이퍼 파라미터를 찾은 경우, 기존의 알고리즘들 대비 정확도가 높고, 동적으로 feature map 이미지의 크기도 조절가능하다는 장점이 있다. GAF 또한 98~99%의 높은 정확도를 보이지만, feature map 이미지의 크기를 동적으로 조절할 수 없어 크다는 단점이 존재한다.

▸ **주제어**: 시계열 데이터, GAF, MTF, RP, STFT, 스펙트로그램, CNN

• First Author: Sung-Kyu Hong, Corresponding Author: Sang-Chul Kim
*Sung-Kyu Hong (hsung951027@gmail.com), Graduate School of Software Technology, Kookmin University
**Sang-Chul Kim (sckim7@kookmin.ac.kr), School of Computer Science, Kookmin University
• Received: 2023. 01. 16, Revised: 2023. 04. 13, Accepted: 2023. 04. 17.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

시계열 예측 (Time Series Classification) 문제는 데이터 마이닝 분야에서 중요하고 어려운 문제로 꼽히고 있다. 시계열 데이터의 양이 늘어날수록 많은 시계열 예측 알고리즘들은 제안되고 있다[1].

다양한 연구 및 기술 개발이 활발히 이루어짐에 따라 딥러닝 기반의 접근 방식 또한 시계열 데이터에 관련된 태스크에도 적용되고 있다. 그중 대표적으로 이미지인식 및 영상처리 분야에서 뛰어난 성능을 보이는 CNN (Convolutional Neural Network) 이 있다. 그뿐만 아니라 시계열 데이터에 적합한 RNN (Recurrent Neural Network) 기반의 모델들도 있다[2].

그러나, 시계열 데이터 예측 (Time Series Forecasting) 태스크에서는 RNN을 많이 적용하는 반면, 시계열 데이터 분류 (Time Series Classification) 태스크에는 다음과 같은 고질적인 문제가 존재한다[3]. (1) RNN 기반의 신경망 구조는 시간 순서에 따라 각 원소를 예측하도록 설계되어 있다. (2) 시계열 데이터의 길이가 길어지면 그레디언트 소실문제가 고질적으로 발생한다. (3) RNN은 계산적인 이유로 학습과 병렬화가 어렵다[4]. 반면, CNN은 커널 (Kernel)을 통해 주변 정보를 고려하며, 병렬적으로 채널에 대한 연산을 수행하여 Feature map을 형성하므로 RNN의 고질적인 문제를 보완하기 위해 시계열 데이터 분류 문제에 대해서는 CNN이 적용되는 사례들이 있다.

시계열 데이터를 CNN에 적용하기 위해 시계열 데이터를 행렬화하는 다양한 시도들이 이루어진다. 이를테면 시계열 데이터의 특징들을 추출하여 이미지로 변환하기 위한 GAF (Gramian Angular Field), MTF (Markov Transition Field), RP (Recurrence Plot) 등 다양한 알고리즘들이 존재한다. 본 논문에서는 푸리에 변환을 신호 분야에 국한하지 않고, 일반적인 시계열 데이터에 적용한다. 그리고, STFT를 통해 시계열 데이터를 스펙트로그램 이미지 형태로 변환하고, 앞서 언급한 세 가지 알고리즘들을 통해 변환된 이미지들을 CNN로 분류한 성능을 비교 및 분석한다.

본 논문의 구성은 다음과 같다. 1장에서는 연구 배경과 목적을 설명하고, 2장에서는 실험을 진행하기 위한 시계열 데이터 이미지화 알고리즘 및 합성곱 신경망에 대한 이론적인 배경을 설명한다. 3장에서는 이론적인 배경을 바탕으로 하이퍼 파라미터를 수정하며 실행 진행 과정을 설명하고, 4장에서는 정확도(Accuracy) 지표를 기반으로 분류 성능에 대한 결과를 분석한다. 5장에서는 4장의 결과를 바탕으로 결론과 6장은 향후 연구 방향을 제시한다.

II. Preliminaries

1. Fourier Transform

1.1 Fourier Transform

Fourier Transform은 주로 신호 데이터에서 많이 활용된다. 예를 들면, 원본 신호를 주기 신호 성분으로 분류하여 백색 소음과 같은 노이즈를 제거하는 데에 유용하다. 여기서 말하는 주기 신호는 다음 오일러 공식, $e^{j2\pi ux} = cos(2\pi ux) + j \times sin(2\pi ux)$ 에 사용된 sin, cos과 같은 정현파를 의미한다. 오일러 공식을 통해 sin, cos과 같은 주기함수를 복소 정현파의 형태로 식 (1)과 같이 쉽게 표현할 수 있다.

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx$$
$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du \quad (1)$$

식 (1)의 $F(u)$는 원본 입력 신호, $f(x)$는 각 복소 정현파의 강도(가중치)를 의미한다. 위 식은 푸리에 급수 공식으로부터 극한을 적용하여 유도할 수 있다.

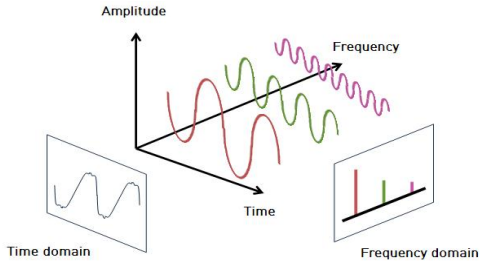

Fig. 1. Fourier transform

하지만, Fig. 1에서처럼 Time domain에서 Frequency domain으로 변환되기 때문에 원본 신호가 시간대별로 주파수가 다른 경우, 시간대별 주파수 성분의 특징을 파악하기 어렵다.

1.2 Fourier Transform

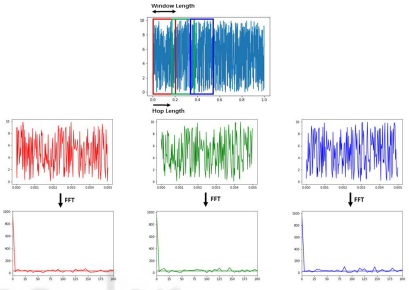

Fig. 2. Short time Fourier transform

Fourier Transform의 한계점은 원본 신호가 시간대별로 주파수가 달라질 경우, 시간대별 주파수와 진폭의 특성을 파악하기가 힘들다는 단점이 존재한다. Fig. 2의 Short Time Fourier Transform은 이러한 단점을 극복하기 위해 슬라이딩 윈도우 방식 (시간대를 window 크기의 단위로 hop length만큼 이동)을 적용하여 윈도우 단위로 Discrete Time Fourier Transform을 수행한다. Discrete Time Fourier Transform은 연속 신호로부터 샘플링 과정을 거쳐 이산 신호로 변환한 다음, Fourier Transform을 적용하는 것을 의미한다.

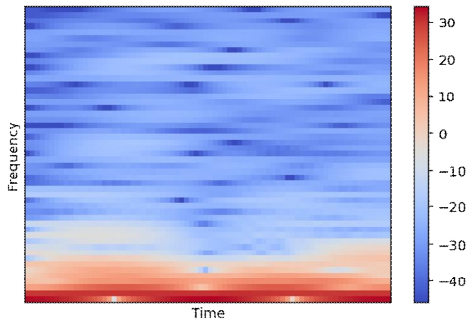

Fig. 3. spectrogram

특정 신호를 Short Time Fourier Transform의 결괏값에 절댓값을 취하고, 데시벨 스케일로 변환한 스펙트로그램을 Fig. 3에 나타내었다. 스펙트로그램의 적용 사례로는 소리 혹은 생체 신호 등을 변환한 감성 분류 작업이 있다[5].

2. Imaging algorithm for time series data

2.1 GAF (Gramian Angular Field)

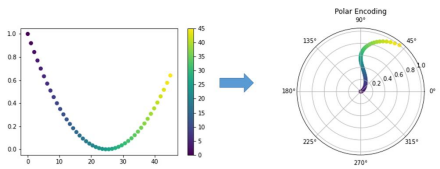

Fig. 4. Polar coordinate encoding

Gramian Angular Field는 식 (2)의 극좌표계를 통해 시계열 데이터를 표현하는 방법이다[6].

$$X = \{x_1, x_2, ..., x_n\}$$
$$\begin{cases} \Theta = arccos(x_i) & \text{where } -1 \le x_i \le 1 \\ r = \frac{i}{N} \end{cases} \quad (2)$$

시계열 데이터 X 는 n 개의 타임 스텝으로 구성되었다고 가정한다. 데이터가 너무 커지는 것을 방지하기 위해 시계열 데이터에 Min-Max scale을 취한 후, Fig. 4의 극좌표계로 변환한다. 이때, 각도는 시계열 데이터값에 arccos을 취하고, 반지름은 i 번째 타임 스텝에 정규화를 위해 시계열 데이터의 길이로 나누어준다.

$$G = \begin{bmatrix} cos(\Theta_1 + \Theta_1) & cos(\Theta_1 + \Theta_2) & \cdots & cos(\Theta_1 + \Theta_n) \\ cos(\Theta_2 + \Theta_1) & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ cos(\Theta_n + \Theta_1) & \cdots & \cdots & cos(\Theta_n + \Theta_n) \end{bmatrix} \quad (3)$$

$$G = \begin{bmatrix} sin(\theta_1 - \theta_1) & sin(\theta_1 - \theta_2) & \cdots & sin(\theta_1 - \theta_n) \\ sin(\theta_2 - \theta_1) & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ sin(\theta_n - \theta_1) & & & sin(\theta_n - \theta_n) \end{bmatrix} \quad (4)$$

최종적으로 Gramian Angular Field는 Gramian Matrix를 통해 표현된다. 식 (3)의 GASF (Gramian Angular Summation Field)는 각 원소는 극좌표로 변환된 벡터 간의 각도 합에 cos을 취한 값이다. 반대로 식(4)의 GADF (Gramian Angular Difference Field)는 벡터 간의 각도 차에 sin을 취한 값이다.

GAF의 장점은 다음과 같다. (1) 좌측 상단에서 우측 하단으로 갈수록 시간이 증가하여 시간 종속성을 유지할 수 있다. (2) 주대각선 성분은 원본 데이터를 포함하고 있어 심층 신경망으로 인해 학습되는 고차원의 피처들로부터 시계열 데이터를 재구성할 수 있다[7].

2.2 MTF (Markov Transition Field)

Markov Transition Field는 시간 도메인의 정보를 보존하기 위해 마르코프 전이 확률들을 표현한 방법이다[6].

$$W = \begin{bmatrix} p_{1,1}|P(x_{t-1} \in q_1 | x_t \in q_1) & p_{1,2}|P(x_{t-1} \in q_1 | x_t \in q_2) & \cdots & p_{1,Q}|P(x_{t-1} \in q_1 | x_t \in q_Q) \\ p_{2,1}|P(x_{t-1} \in q_2 | x_t \in q_1) & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ p_{Q,1}|P(x_{t-1} \in q_Q | x_t \in q_1) & \cdots & \cdots & p_{Q,Q}|P(x_{t-1} \in q_Q | x_t \in q_Q) \end{bmatrix} \quad (5)$$

먼저 시계열 데이터를 Q개의 구간(Bin)으로 나눈 뒤에, 각 시계열 데이터값에 해당되는 구간을 할당한다. 그리고, $Q \times Q$ 크기의 마르코프 전이 행렬을 구성한다. 식 (5)의 행렬 W의 각 원소는 이전 타임 스텝의 데이터가 속하는 구간에서 다음 타임 스텝의 데이터가 속하는 구간으로 전이하는 빈도수를 의미한다.

그 후, 행렬의 열의 합을 1로 정규화하여 구간별 전이 확률로 마르코프 전이 행렬을 구성할 수 있다. 이 과정은

시계열 데이터 X의 분포와 시간 종속성에 대해 robust할 수 있지만, 많은 정보 손실을 유발한다.

$$M = \begin{bmatrix} p_{i,j}|x_1 \in q_i, x_1 \in q_j & p_{i,j}|x_1 \in q_i, x_2 \in q_j & \cdots & p_{i,j}|x_1 \in q_i, x_n \in q_j \\ p_{i,j}|x_2 \in q_i, x_1 \in q_j & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ p_{i,j}|x_n \in q_i, x_1 \in q_j & \cdots & \cdots & p_{i,j}|x_n \in q_i, x_n \in q_j \end{bmatrix} \quad (6)$$

이러한 정보 손실을 극복하기 위해 $n \times n$ 행렬을 재구성할 수 있다. 식 (6)에 표시한 행렬 M의 원소는 i 번째 타입 스텝 데이터의 구간에서 j 번째 타임 스텝 데이터 구간으로 전이할 확률을 나타낸다.

2.3 RP (Reccurrence Plot)

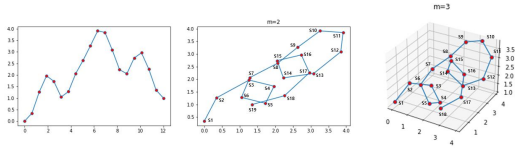

Fig. 5. Trajectory encoding

Recurrence Plot은 시계열 데이터에 대하여 회귀를 2차원 표현을 통해 m차원 위상 공간 궤도를 탐색하는 것을 목표로 하는 시각화 도구이다. Fig. 5에 표시한 회귀는 궤도가 이전의 위치로 돌아오는데 걸리는 시간을 의미한다[8].

$\overrightarrow{s_n} = (s_n, s_{n+\tau}, \cdots, s_{n+(m-1)\tau})$ 식은 s_n이 n번째 타임 스텝의 시계열 데이터를 의미할 때, 시계열 데이터의 궤도를 m차원 공간에 투영한 벡터의 일반식이다. τ는 시간 지연 (time delay)를 의미하고, m은 투영할 벡터 공간의 차원 수이다.

예를 들어 $\overrightarrow{s_n} = (s_n, s_{n+1})$ $R_{ij} = dist(s_i, s_j) = \Theta(\varepsilon - ||\overrightarrow{s_i} - \overrightarrow{s_j}||)$ 에서 τ는 1, m이 2인 경우 시계열 데이터의 궤도를 오른쪽의 2차원 공간에 표현한 것이다. 즉, 벡터 s_n은 타임 스텝 n에서 타임 스텝 $n+1$시점으로 이동하는 궤적을 2차원 공간에 투영한 벡터이다.

$$R = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1j} \\ R_{21} & \cdots & \cdots & R_{2j} \\ \cdots & \cdots & \cdots & \cdots \\ R_{i1} & R_{i2} & \cdots & R_{ij} \end{bmatrix} \quad (7)$$

식 (7)은 Recurrence Plot을 의미하고, 식은 Recurrence Plot의 원소로 앞에서 표현한 궤적을 거리를 표현한 것이다. ϵ은 거리 행렬에 반영하기 위한 임계치이고, θ는 활성화 함수를 의미한다. 행렬의 i 번째 행의 j번째 요소 R_{ij}는 궤도의 i 번째 벡터 s_i와 j 번째 벡터 s_j사이의 거리를 나타낸 것이다.

$$R_{i,j} = \begin{cases} D & \text{if } D = dist(s_i, s_j) \geq \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

식 (8)의 Const-RP는 임계치 ϵ을 반영하지 않는 Recurrence Plot이다. Bin-RP는 Binary Recurrence Plot의 약자로, 활성화 함수 θ를 Heaviside function으로 적용하여 임계치 ϵ보다 작으면 0, 크면 1로 값을 반영한다. ReLU-RP는 활성화 함수 θ를 ReLU function으로 적용하여 Fig. 13 수식처럼 표현할 수 있다[9].

3. UCR time series archive

UCR time series archive는 UCI Archive로부터 영감을 받아 2002년에 소개된 time series data mining community에서 중요한 리소스이다[10]. 처음에는 16개의 데이터셋으로 구성되었지만 시간이 지남에 따라 데이터셋의 개수는 확장되었다. 2015년에 85개까지 데이터셋이 확장되었다.

GunPoint 데이터셋은 학습 데이터 샘플의 개수는 50개, 테스트 데이터 샘플의 개수는 150개다. GunPoint 데이터셋의 레이블은 “Gun”과 “Point” 두 가지 시나리오로 나누어진다. 각각의 시나리오의 배우는 눈높이까지 타겟을 조준한다. 이때 배우의 손의 x축 좌표를 시계열 데이터로 기록하였다. 각각의 시나리오에 대한 영상들은 삼성 갤럭시 8로 촬영하였으며 초당 30프레임으로 5초 주기이다.

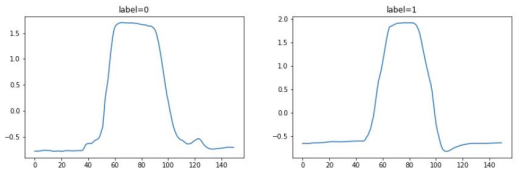

Fig. 6. GunPoint dataset sample

Fig. 6의 “Gun” 시나리오(label=0)는 배우가 가짜 총을 집고 타켓을 조준한다. 그 다음 총을 허리의 총대에 넣고 손을 시작 위치로 가져다 놓으면 해당 시나리오는 끝이 난다. 반면, “Point” 시나리오(label=1)는 허리에 총대가 없기 때문에 손이 올라갔다가 바로 내려간다.

III. The Proposed Scheme

1. Application of imaging algorithm for time series data

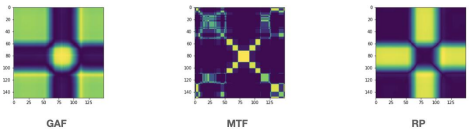

Fig. 7. Time series images

시계열 데이터의 이미지화를 위해 pyts 패키지를 사용한다. Fig. 7은 특정 시계열 데이터에 대해 GAF, MTF, RP를 각각 적용한 2D 이미지이다. 각각의 이미지화 기법별로 적용할 수 있는 하이퍼 파라미터가 존재한다. 별개로 Fig. 8에 나타낸 스펙트로그램 이미지화를 위해 numpy 패키지와 librosa 패키지를 이용했다. librosa 패키지를 통해 stft() 메소드를 호출하면 복소수를 반환한다. 이를 np.abs() 메소드를 통해 절대값으로 변환한 다음, amplitude_to_db() 메소드를 통해 데시벨 스케일로 변환했다.

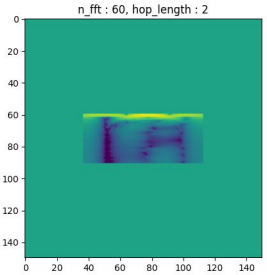

Fig. 8. Spectrogram

2. Hyperparameter tuning of imaging algorithm for time series data

GramianAngularField 클래스의 생성자 파라미터는 'method'로 GAF 행렬의 원소 구성 방식을 선택할 수 있다. 'summaion'을 전달할 경우 벡터 간의 각도 합을 $\cos(\theta_1+\theta_2)$로 원소를 구성하므로 GASF를 추출한다. 반대로 'difference'를 전달할 경우 벡터 간의 각도 차를 $\sin(\theta_1-\theta_2)$로 원소를 구성하여 GADF를 추출한다.

MarkovTransitionField 클래스의 생성자 파라미터는 'n_bins'와 'strategy'가 있다. 'n_bins'는 시계열 데이터를 나눌 구간의 개수를 의미하고, 'strategy'는 구간을 나누는 방법들을 의미한다. 구간을 나눌 방법은 세 가지이다. 첫 번째 방법인 'uniform'은 각 구간의 너비를 동일하게 나눈다. 두 번째 방법인 'quantile'은 모든 구간에 포함되는 데이터 샘플의 수를 동일하게 나눈다. 세 번째 방법인 'normal'은 표준 정규 분포에 따라 구간의 경계선을 결정한다.

RecurrencePlot 클래스의 생성자 파라미터는 'dimension'이 있다. 전달받은 'dimension'의 파라미터가 m이라 가정할 때 m개의 타임 스텝 데이터를 m차원 공간에 투영한다.

시계열 데이터를 스펙트로그램으로 변환하기 위해 librosa 패키지를 사용한다. 먼저 stft() 함수를 통해 Short-Time Fourier Transform의 결괏값을 얻은 후에 절댓값을 취한다. 그 다음 이 값을 amplitude_to_db() 함수의 인자로 전달하면 스펙트로그램을 얻을 수 있다. 이때 stft() 메소드에 조정할 파라미터는 'n_fft'와 'hop_length'로 윈도우의 크기와 이동 거리를 의미한다.

3. Implementation

3.1 Implementation of AlexNet-based neural network

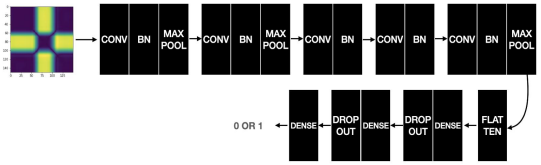

Fig. 9. Scaled AlexNet

본 실험에서는 Fig. 9에서와같이 이미지화 기법별로 성능을 비교하기 위한 CNN모델로 ILSVRC-2012 대회에서 우승한 AlexNet을 참고했다. 기존의 AlexNet은 2개의 GPU를 사용하기 위해 feature map을 병렬적으로 분리했지만, 본 실험에서는 1개의 GPU (RTX 3060 TI 8GB)를 사용하기 때문에 분리하지 않았다. 그리고, 측면 억제 현상 (lateral inhibition)을 방지하기 위해 사용된 Local Response Normalization을 은닉층에서의 입력에 대한 분산이 달라지는 Internal Corvariate Shift 현상을 방지하기 위한 Batch Normalization으로 대체했다[11].

Table 1. CNN architecture

AlexNet	Scaled AlexNet
48 x 11 x 11 Conv, 48 x 11 x 11 Conv	24 x 3 x 3 Conv
128 x 5 x 5 Conv, 128 x 5 x 5 Conv	64 x 3 x 3 Conv
192 x 3 x 3 Conv, 192 x 3 x 3 Conv	96 x 3 x 3 Conv
192 x 3 x 3 Conv, 192 x 3 x 3 Conv	96 x 3 x 3 Conv
128 x 3 x 3 Conv, 128 x 3 x 3 Conv	64 x 3 x 3 Conv
2048 Dense, 2048 Dense	1024 Dense
2048 Dense, 2048 Dense	1024 Dense
1000 Dense (Softmax)	1 Dense (Sigmoid)

기존의 AlexNet은 224 x 224 컬러 이미지 입력이 기준인 신경망이다. 하지만, 실험 데이터 특성상 Gunpoint 데이터셋의 하나의 샘플의 길이가 150 이기 때문에 이미지 크기가 150 x 150 이하이다. 그러므로, 너무 많은 Convolution 필터를 사용할 경우 오버피팅이 발생하여 기존의 AlexNet 보다 적은 수의 파라미터를 유지하도록 Table 1처럼 구조를 변형했다. 추가로, Gunpoint 데이터셋의 시나리오는 두 개로써 이진 분류 문제에 해당하기 때문에 마지막 Dense 레이어에서 시그모이드 활성화 함수를 통해 결과를 도출한다.

```python
import tensorflow.keras as keras


class ConvolutionalBlock(keras.layers.Layer):
    def __init__(self, filters: int, kernel_length: int, stride: int,
                 max_pool_length=None, max_pool_stride=None, padding='valid', **kwargs):
        super().__init__(**kwargs)
        self.block = [
            keras.layers.Conv2D(filters=filters,
                                kernel_size=(kernel_length, kernel_length),
                                kernel_initializer=keras.initializers.he_normal(1000),
                                strides=(stride, stride),
                                activation='relu',
                                padding=padding),
            keras.layers.BatchNormalization()
        ]
        if max_pool_length:
            self.block.append(
            keras.layers.MaxPool2D(pool_size=(max_pool_length, max_pool_length),
                                   strides=(max_pool_stride, max_pool_stride)))

    def call(self, inputs, **kwargs):
        for layer in self.block:
            inputs = layer(inputs)
        return inputs


class ScaledAlexNet(keras.Model):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.blocks = [
            ConvolutionalBlock(24, 3, 1, 3, 2),
            ConvolutionalBlock(64, 3, 1, 3, 2, 'same'),
            ConvolutionalBlock(96, 3, 1, padding='same'),
            ConvolutionalBlock(96, 3, 1, padding='same'),
            ConvolutionalBlock(64, 3, 1, 3, 2, 'same'),
            keras.layers.Flatten(),
            keras.layers.Dense(1024, activation='relu'),
            keras.layers.Dense(1024, activation='relu'),
            keras.layers.Dense(1, activation='sigmoid')
        ]

    def call(self, inputs, training=None, mask=None):
        for block in self.blocks:
            inputs = block(inputs)
        return inputs
```

Fig. 10. Scaled AlexNet Implementation

Fig. 10에서와같이 신경망 구현을 위해 Tensorflow 2.6.0 버전을 사용하였으며, Keras API를 사용했다. 중복되는 코드를 제거하기 위해 반복되는 패턴은 ConvolutionalBlock 클래스로 사전 정의했다.

3.2 AlexNet-based neural network training

본 논문에서는 이진 분류 (Binary Classification)에 대한 실험을 진행하며, 지표(Metric)는 정확도 (Accuracy)를 사용하여 각 이미지화 알고리즘의 하이퍼 파라미터를 수정하며 Scaled AlexNet을 학습한다. 최적화 (Optimization)을 위해 모멘텀 (Momentum) 옵티마이저를 사용하였으며, learning rate은 0.0005, momentum은 0.5를 지정했다. 손실함수 (Loss function)은 binary cross entropy를 사용하여 75번의 epochs를 통해 반복 학습을 했다.

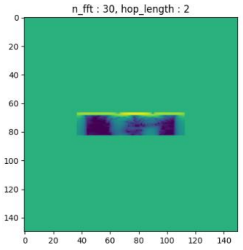

Fig. 11. Zero Padding

각 이미지화 알고리즘의 하이퍼 파라미터 수정 시 이미지의 크기가 변하는 경우, 기존의 시계열 데이터의 길이보다 짧은 너비와 높이가 나타나 이는 Fig. 11의 제로 패딩 (Zero Padding)을 통해 이미지의 크기를 통일하여 동일한 신경망으로 학습을 진행하였다.

```python
from multiprocessing import Process, Queue


def run_stft_with_params(evaluator, params, q: Queue):
    for n_epochs, n_fft, hop_length in params:
        _, _, accuracy = evaluator.process_stft(n_epochs, n_fft, hop_length)
        q.put(accuracy)
    return


def monte_carlo_simulate_stft(evaluator, n_epochs):
    q = Queue()
    params = list()
    n_ffts = [i for i in range(30, 121)]
    hop_lengths = [2, 3, 4]

    for hop_length in hop_lengths:
        for i, n_fft in enumerate(n_ffts):
            params.extend([(n_epochs, n_fft, hop_length)])
            if i != 0 and i % 10 == 0:
                p = Process(target=run_stft_with_params, args=(evaluator, params, q))
                p.start()
                p.join()
                params.clear()
```

Fig. 12. Resolving graphics memory issues using processes from the multiprocessing module

STFT 하이퍼 파라미터 튜닝 과정에서 좀 더 확실한 인사이트를 얻기 위해 비교적 많은 270가지의 경우의 수에 대해 학습을 진행했다. 해당 과정에서 발생한 그래픽 카드 메모리 부족 문제를 해결하기 위해 multiprocessing 모듈을 이용했다. Fig. 12에서와같이 10가지 경우를 주기로 새로운 프로세스를 생성하여 CNN 모델을 학습한 뒤, 프로세스 간 통신이 가능한 큐(Queue)에 해당 프로세스는 정확도만 추출한 후 학습을 전부 마친 프로세스는 제거하여 그래픽 카드의 메모리 점유 문제를 해결하였다.

IV. Experiments

1. GunPoint dataset accuracy

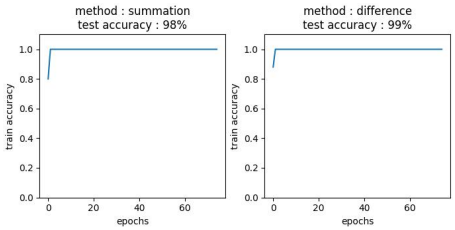

Fig. 13. GAF result

Fig. 13은 GAF의 하이퍼 파라미터를 변경하여 성능을 비교한 그래프이다. ‘method’ 파라미터에 ‘summation’과 ‘difference’를 둘 다 전달하더라도 분류 성능에 크게 차이가 나지 않는다. 단순히 극좌표계로 변환된 시계열 데이터의 각 타임 스텝의 데이터 벡터 간의 상관관계를 표현하는 방법의 차이이기 때문이다.

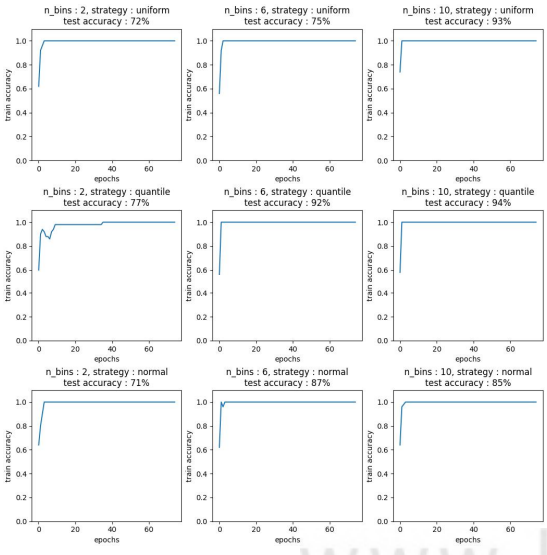

Fig. 14. MTF result

Fig. 14는 MTF의 하이퍼 파라미터를 변경하여 성능을 비교한 그래프이다. ‘strategy’ 파라미터에 ‘uniform’과 ‘quantile’, ‘normal’를, ’n_bins’ 파라미터에 2, 6, 10을 전달했다. 어떠한 ‘strategy’를 사용하든 데이터의 분포를 나누는 구간을 의미하는 ’n_bins’ 파라미터에 전달되는 인자의 크기가 늘어날수록 높은 정확도를 보여주었다. 이는 데이터들이 포함되는 구간의 개수가 늘어남에 따라 섬세한 표현이 가능하기 때문이다. 단, MTF의 normal strategy는 bin을 나누는 기준이 표준 정규 분포의 사분위수(quantile)로 나눈다. 데이터들의 값들의 분포를 표준 정규 분포로 스케일링하기 위해 sklearn 패키지의 StandardScaler 클래스를 이용했다.

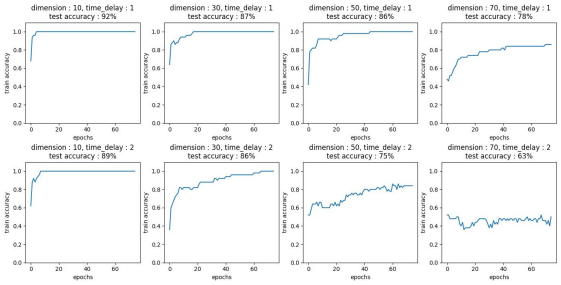

Fig. 15. RP result

Fig. 15는 RP의 하이퍼 파라미터를 변경하여 성능을 비교한 그래프이다. ‘time_delay’ 파라미터에는 1과 2를 전달하고, ‘dimension’ 파라미터에는 10, 30, 50, 70을 전달했다. ‘dimension’ 파라미터에 전달되는 인자의 크기가 커질수록 고차원을 한 점에 정보를 요약하기 때문에, 절대적인 데이터의 크기(거리 행렬의 크기)는 줄어들지만, 분류 성능이 저하된다. 그리고, ‘time_delay’의 파라미터가 커지면 거리를 파악하고자 하는 궤적의 시간 간격이 넓어지면서 분류 성능이 저조해지는 것을 확인할 수 있다.

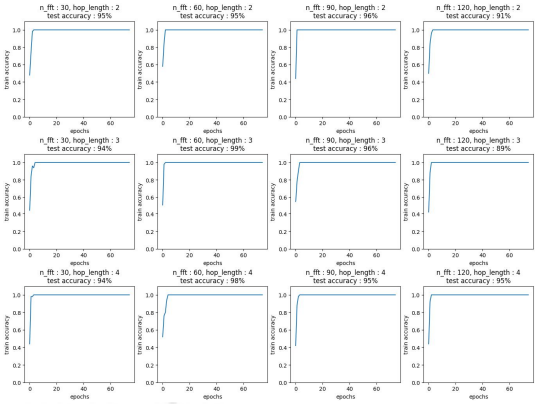

Fig. 16. STFT result

Fig. 16은 STFT의 하이퍼 파라미터를 변경하여 성능을 비교한 그래프이다. 'hop_length' 파라미터에는 2, 3, 4를 전달하고, 'n_fft' 파라미터에는 30, 60, 90, 120을 전달했다. 'n_fft' 파라미터가 커질수록 푸리에 변환을 하는 윈도우의 크기가 늘어나 일정 수준까지는 성능이 향상되다가 그 이후로는 정확도가 저조해지는 것을 확인할 수 있다.

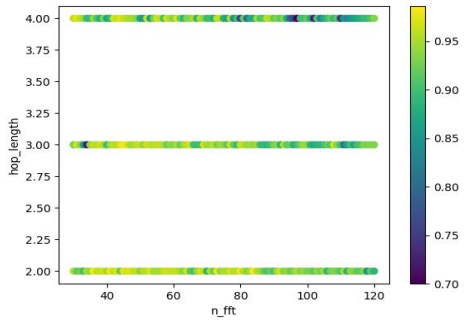

Fig. 17. STFT meshgrid

좀 더 정확한 결과를 도출하기 위해 n_fft를 30부터 120사이의 모든 값을 대입하여 270가지의 경우의 스펙트로그램 데이터셋을 생성한 뒤, CNN 모델을 학습시켰다. Fig.17은 STFT meshgrid를 나타내며, 세로축은 n_fft, 가로축은 hop_length를 의미한다. 그리고 각각의 meshgrid의 포인트들의 밝기는 정확도를 의미한다. 그림을 보면 hop_length가 높아질수록 어두운 포인트의 빈도가 높아진다. 그리고 n_fft 40~60을 기준으로 멀어질수록 어두워지는 모습을 확인할 수 있다.

2. Other dataset evaluation

앞서 GunPoint 데이터셋을 통해 하이퍼 파라미터 조정에 따른 정확도 양상을 파악하였다. 하지만, 좀 더 확실한 결론을 위해 추가로 다른 데이터셋을 통해 실험을 진행했다. 실험을 위해 진행한 데이터셋들의 정보는 Table 2와 같다.

Table 2. Training datasets

Name	classes	time stamps	train size	test size	type
GunPoint	2	150	50	150	MOTION
GunPointAge Span	2	150	135	316	MOTION
GunPointMale VersusFemale	2	150	135	316	MOTION
GunPointOld VersusYoung	2	150	135	316	MOTION
PowerCons	2	144	180	180	DEVICE

각 데이터셋의 하이퍼 파라미터 조정은 GunPoint 데이터셋에 대한 하이퍼 파라미터 조정과 동일하게 수행하였다. 각 데이터별로 하이퍼 파라미터를 조정한 모든 경우에 대해 정확도를 Table 3 에서 Table 22까지 나열했다. 굵은 표시는 해당 표에서의 최고 수치를 의미한다.

3.1 GunPoint dataset

Table 3. GunPoint GAF Test Accuracy

GAF	Summation method	Difference method
	98%	**99%**

Table 4. GunPoint MTF Test Accuracy

MTF	n_bins=2	n_bins=6	n_bins=10
strategy=uniform	72%	75%	93%
strategy=quantile	77%	92%	**94%**
strategy=normal	71%	87%	85%

Table 5. GunPoint RP Test Accuracy

RP	dimension=10	dimension=30	dimension=50
time_delay=1	**92%**	87%	86%
time_delay=2	89%	86%	75%
strategy=normal	71%	87%	85%

Table 6. GunPoint STFT(Spectrogram) Accuracy

STFT	n_fft=30	n_fft=60	n_fft=90	n_fft=120
hop_length=2	95%	95%	96%	91%
hop_length=3	94%	**99%**	96%	89%
hop_length=4	94%	98%	95%	95%

3.2 GunPointAgeSpan dataset metrics

Table 7. GunPointAgeSpan GAF Test Accuracy

GAF	Summation method	Difference method
	95%	**95%**

Table 8. GunPointAgeSpan MTF Test Accuracy

MTF	n_bins=2	n_bins=6	n_bins=10
strategy=uniform	84%	87%	**92%**
strategy=quantile	81%	88%	88%
strategy=normal	59%	81%	84%

Table 9. GunPointAgeSpan RP Test Accuracy

RP	dimension=10	dimension=30	dimension=50
time_delay=1	**90%**	79%	75%
time_delay=2	89%	74%	54%

Table 10. GunPointAgeSpan STFT(Spectrogram) Accuracy

STFT	n_fft=30	n_fft=60	n_fft=90	n_fft=120
hop_length=2	91%	49%	59%	59%
hop_length=3	**94%**	49%	59%	50%
hop_length=4	**94%**	58%	57%	58%

3.3 GunPointMaleVersusFemale dataset metrics

Table 11. GunPointMaleVersusFemale GAF Test Accuracy

GAF	Summation method	Difference method
	100%	99%

Table 12. GunPointMaleVersusFemale MTF Test Accuracy

MTF	n_bins=2	n_bins=6	n_bins=10
strategy=uniform	96%	**99%**	97%
strategy=quantile	81%	92%	94%
strategy=normal	47%	91%	89%

Table 13. GunPointMaleVersusFemale RP Test Accuracy

RP	dimension =10	dimension =30	dimension =50	dimension =70
time_delay=1	**99%**	**99%**	**99%**	**99%**
time_delay=2	**99%**	**99%**	75%	58%

Table 14. GunPointMaleVersusFemale STFT(Spectrogram) Accuracy

STFT	n_fft=30	n_fft=60	n_fft=90	n_fft=120
hop_length=2	**98%**	77%	58%	62%
hop_length=3	97%	67%	53%	59%
hop_length=4	**98%**	61%	51%	50%

3.4 GunPointOldVersusYoung dataset metrics

Table 15. GunPointOldVersusYoung GAF Test Accuracy

GAF	Summation method	Difference method
	95%	**94%**

Table 16. GunPointOldVersusYoung MTF Test Accuracy

MTF	n_bins=2	n_bins=6	n_bins=10
strategy=uniform	81%	92%	**93%**
strategy=quantile	81%	91%	90%
strategy=normal	54%	91%	**93%**

Table 17. GunPointOldVersusYoung RP Test Accuracy

RP	dimension =10	dimension =30	dimension =50	dimension =70
time_delay=1	95%	93%	95%	89%
time_delay=2	**96%**	95%	60%	52%

Table 18. GunPointOldVersusYoung STFT(Spectrogram) Accuracy

STFT	n_fft=30	n_fft=60	n_fft=90	n_fft=120
hop_length=2	93%	64%	52%	52%
hop_length=3	**99%**	98%	52%	52%
hop_length=4	**99%**	98%	52%	52%

3.5 PowerCons dataset metrics

Table 19. PowerCons GAF Test Accuracy

GAF	Summation method	Difference method
	96%	**94%**

Table 20. PowerCons MTF Test Accuracy

MTF	n_bins=2	n_bins=6	n_bins=10
strategy=uniform	91%	91%	**92%**
strategy=quantile	87%	89%	91%
strategy=normal	84%	81%	85%

Table 21. PowerCons RP Test Accuracy

RP	dimension =10	dimension =30	dimension =50	dimension =70
time_delay=1	97%	**98%**	94%	91%
time_delay=2	96%	92%	94%	63%

Table 22. PowerCons STFT(Spectrogram) Accuracy

STFT	n_fft=30	n_fft=60	n_fft=90	n_fft=120
hop_length=2	91%	88%	89%	86%
hop_length=3	**92%**	88%	90%	91%
hop_length=4	91%	**92%**	85%	90%

V. Conclusions

단변량 시계열 데이터셋을 기준으로 대체적으로 우수한 분류 성능을 보여준 알고리즘은 GAF이다. 하지만, GAF와 MTF 알고리즘 특성상 자체적으로 시계열 데이터를 압축할 방법이 없으므로, GAF는 PAA(Piecewise Aggregation Approximation)를 적용하거나 [12] MTF는 blurring kerenel을 적용하여 인접 픽셀들의 평균을 적용하여 이미지 사이즈를 줄이는 사례가 있다[7]. 반면, RP는 알고리즘의 특성상 각 타임 스텝의 데이터를 투영할 벡터 공간의 차원을 높여 이미지 사이즈를 줄일 수 있는 방법이

존재하지만, 투영할 차원을 높이거나 시간 지연이 높아질수록 정확도가 떨어진다.

GAF 또한 높은 정확도를 보여주었지만, 하이퍼 파라미터를 통해 feature map의 크기를 조절할 수 없다는 단점이 존재한다. MTF 알고리즘의 경우, 구간별로 데이터를 나누기 때문에 시계열 데이터의 분포의 종속성이 강하다. 추가로, 하이퍼 파라미터의 변화에 따라 정확도가 민감하게 반응하는 단점이 존재한다. RP 알고리즘은 극단적인 하이퍼 파라미터의 변화를 주지 않는 다는 전재에서 안정적인 정확도를 보여주었지만, 최대 정확도가 STFT 알고리즘에 미치지 못한 경우가 과반수였다. STFT를 사용할 경우 최적의 하이퍼 파라미터를 전재로 다른 알고리즘에 비해 높은 정확도를 보여주었다. 하지만, 윈도우 사이즈(n_fft)를 높일수록 정확도가 하락했다.

ACKNOWLEDGEMENT

This work was carried out as a result of the research result of the MIST(Ministry of Science, ICT), Korea, under the National Program for Excellence in SW), supervised by the IITP(Institute of Information & communications Technology Planning & Evaluation) in 2022"(2022-0-00964), and the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2023-2018-0-01396) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation. This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ICAN(ICT Challenge and Advanced Network of HRD) program (IITP-2023-2020-0-018260201001) supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation.

REFERENCES

[1] R. M. Samant, M. R. Bachute, S. Gite and K. Kotecha, "Framework for Deep Learning-Based Language Models Using Multi-Task Learning in Natural Language Understanding: A Systematic Literature Review and Future Directions," IEEE Access, vol. 10, pp. 17078-17097, 2022, DOI: 10.1109/ACCESS.2022.3149798.

[2] D. W. Otter, J. R. Medina and J. K. Kalita, "A Survey of the Usages of Deep Learning for Natural Language Processing," in IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 2, pp. 604-624, Feb. 2021, DOI: 10.1109/TNNLS.2020.2979670.

[3] Yang, C.-L.; Chen, Z.-X.; Yang, C.-Y. Sensor Classification Using Convolutional Neural Network by Encoding Multivariate Time Series as Two-Dimensional Colored Images. Sensors 2020, 20, 168. https://doi.org/10.3390/s20010168

[4] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller, "Deep learning for time series classification: a review", Data Mining and Knowledge Discovery, Vol. 33, No. 2, pp. 1-53, July 2019, DOI:10.1007/s10618-019-00619-1

[5] Hee-Jae Lee, David Lee, and Sang-Goog Lee, "2D Emotion Classification using Short-Time Fourier Transform of Pupil Size Variation Signals and Convolutional Neural Network", Journal of Korea Multimedia Society Vol. 20, No. 10, pp. 1646-1654, October 2017, https://doi.org/10.9717/kmms.2017.20.10.1646

[6] Zhiguang Wang and Tim Oates, “Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Network”, Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 2015, https://www.researchgate.net/publication/275970614

[7] Zhiguang Wang and Time Oates, “Imaging Time-Series to Improve Classification and Imputation”, Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, pp. 3939-3945, 2015, https://www.researchgate.net/publication/277603742

[8] Debayle Johan, Hatami Nima, and Gavet Yann “Classification of Time-Series Images Using Deep Convolutional Neural Networks”, Tenth International Conference on Machine Vision, pp. 23. April 2018, DOI:10.1117/12.2309486

[9] Dongho Seo, Junil Ahn, and Haewoon Nam, "Analysis Performance of Burst Signal Detection Using Recurrence Plot Algorithm", JKICS, 2019, Vol. 44, No.11, pp. 2074-2077, 2019, DOI:10.7840/kics.2019.44.11.2074

[10] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Eamonn Keogh, “The UCR Time Series Classification Archive,” 2018, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/

[11] Sergey Ioffe and Christian Szegedy “Batch Noramlization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, Proceedings of the 32nd International Conference on Machine Learning, Vol. 37, pp. 448-456, 2015, https://dl.acm.org/doi/10.5555/3045118.3045167

[12] Eamonn J. Keogh and Michael J. Pazzani “Scaling up Dynamic

Time Warping for Datamining Applications", Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, August 2000, pp.285-289, DOI:10.1145/347090.347153

Authors

Mr. Hong graduated Artificial Intelligence major from the Graduate School of Software Technology at Korea University Seoul, Korea, in 2022. He is currently an AI Engineer of mysterico. He is interested in NLP, Generation Deep Learning.

Dr. Sang-Chul Kim received the Ph.D. degree in Computer Engineering from Oklahoma State University U.S.A., in 2005. Dr. Kim joined the faculty of the School of Computer Science at Kookmin University, Seoul, Korea, in 2006. He is currently a Professor in the School of Computer Science, Kookmin University. He is interested in Artificial Intelligence internet and mobile computing and cloud computing.