Encoding 1d Time Series data as images

What is Gramian matrix?

$$G(x_1, \ldots, x_n) = \begin{vmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \ldots & \langle x_1, x_n \rangle \\ \langle x_2, x_1 \rangle & \langle x_2, x_2 \rangle & \ldots & \langle x_2, x_n \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x_n, x_1 \rangle & \langle x_n, x_2 \rangle & \ldots & \langle x_n, x_n \rangle \end{vmatrix}.$$

- Gram Matrix of a set of n vectors is a matrix defined by the dot-product of every couple of vectors
- If we assume that all the vectors have magnitude as 1, then;

$$G = \begin{pmatrix} \cos(\phi_{1,1}) & cos(\phi_{1,2}) & \ldots & cos(\phi_{1,n}) \\ \cos(\phi_{2,1}) & cos(\phi_{2,2}) & \ldots & cos(\phi_{2,n}) \\ \vdots & \vdots & \ddots & \vdots \\ \cos(\phi_{n,1}) & cos(\phi_{n,2}) & \ldots & cos(\phi_{n,n}) \end{pmatrix}$$

- Here $ø_{i,j}$ represents the angle between the vectors x_i and x_j.

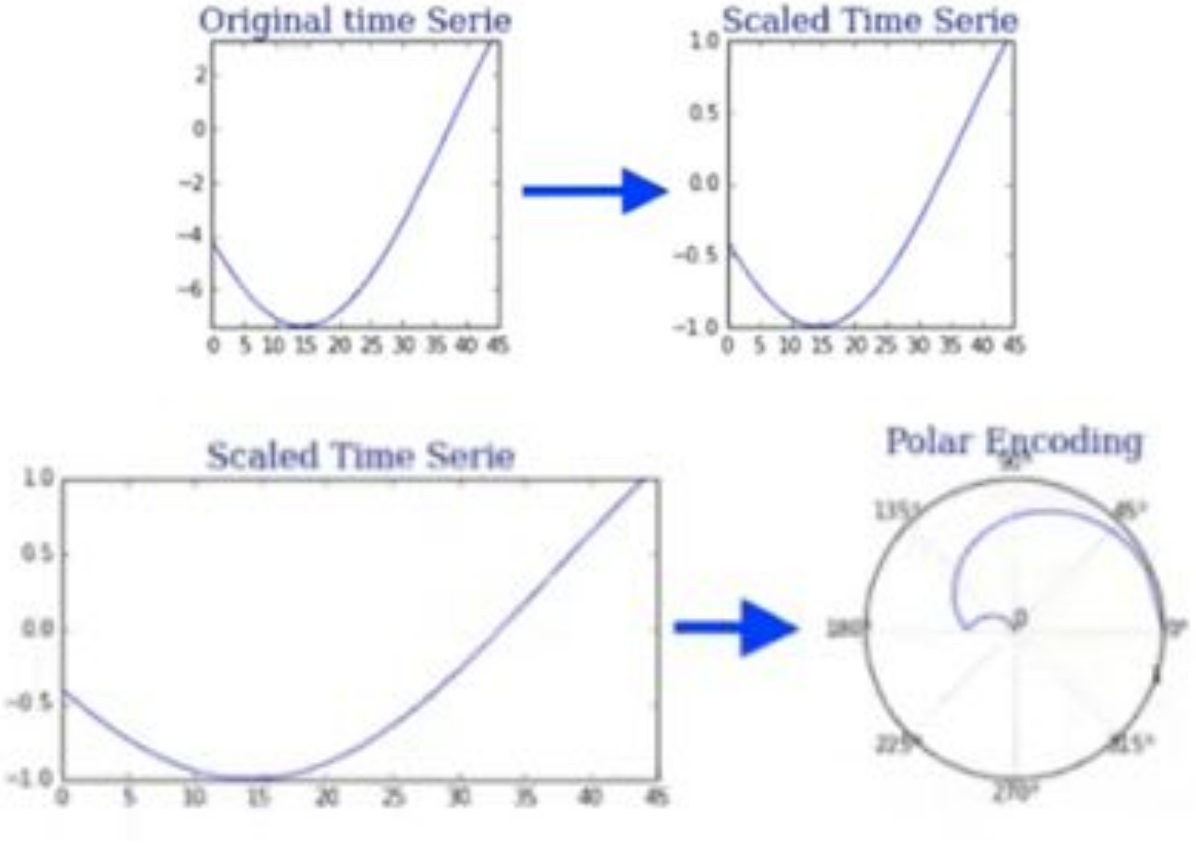
Working of Gramian Angular field

- Input: Time series X = {x1, x2, ..., xn}
- Step-1: Rescale X into the interval [-1, 1] using Min-Max scalar to get *'$\mathcal{X}$_scaled'*

$$\phi_i = \arccos(x_i)$$
$$r_i = \frac{i}{N}$$

- Step-2: Convert *'$\mathcal{X}$_scaled'* to polar coordinates using the transformation given on the right. ($\varnothing_i$, r_i)
- x_i's are taken from X_scaled and as they range from [-1, 1], so $\varnothing_i$ range from [0 to π].
- r_i (where i=1, 2, ... , N) in polar coordinates preserve temporal dependency.

Working of Gramian Angular field



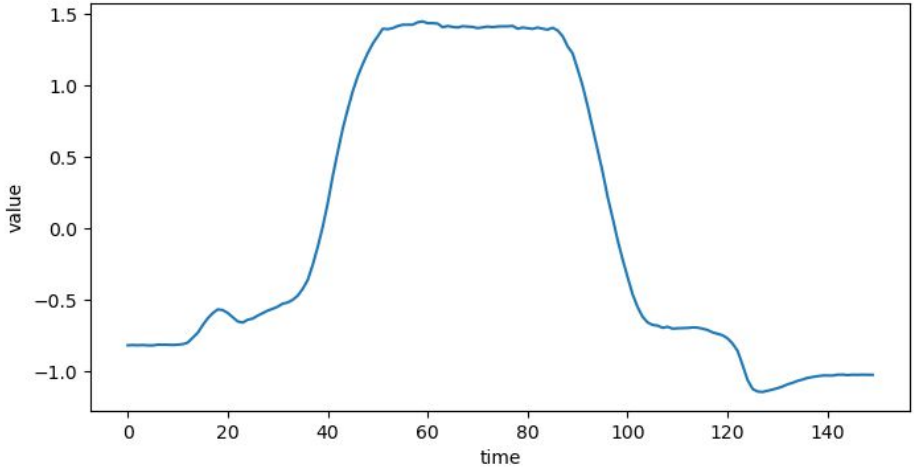
Working of Gramian Angular field

- Step-3: Calculate both Gramian Angular matrices i.e., GASF, GADF.
- GASF- Gramian Angular Summation Field.
- GADF- Gramian Angular Difference Field.
- Save GASF and GADF matrices as images.

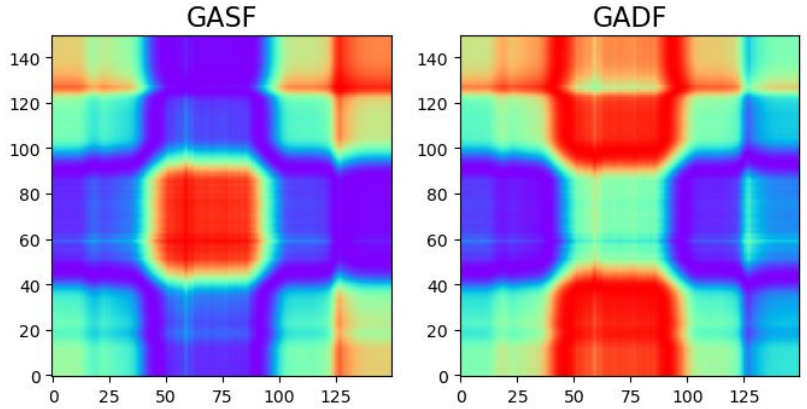
$$\mathbf{GASF} = \begin{bmatrix} \cos(\phi_1 + \phi_1) & \cdots & \cos(\phi_1 + \phi_n) \\ \cos(\phi_2 + \phi_1) & \cdots & \cos(\phi_2 + \phi_n) \\ \vdots & \ddots & \vdots \\ \cos(\phi_n + \phi_1) & \cdots & \cos(\phi_n + \phi_n) \end{bmatrix}$$

$$\mathbf{GADF} = \begin{bmatrix} \sin(\phi_1 - \phi_1) & \cdots & \sin(\phi_1 - \phi_n) \\ \sin(\phi_2 - \phi_1) & \cdots & \sin(\phi_2 - \phi_n) \\ \vdots & \ddots & \vdots \\ \sin(\phi_n - \phi_1) & \cdots & \sin(\phi_n - \phi_n) \end{bmatrix}$$

Outputs from the code written using pyts library



Time series data

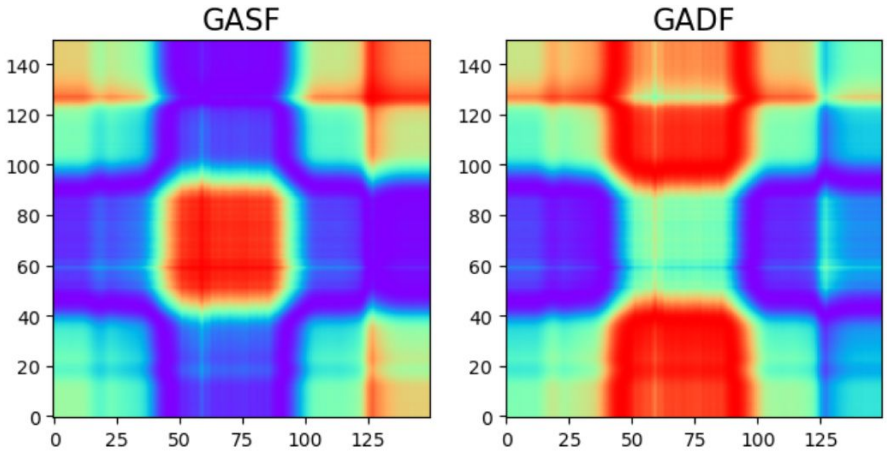


Gramian Angular field

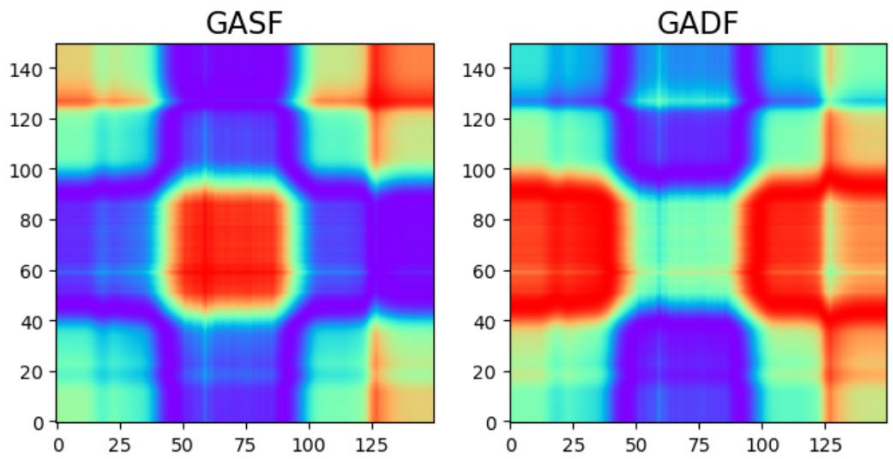
Issue

1. If we use only **GASF**, then for both t1={x1, x2, … , xn} and t2={-x1, -x2, … , -xn} timeseries, we will get same images as output.
    - To avoid this we can use both GASF and GADF and concatenate the. (REF-2)

Example

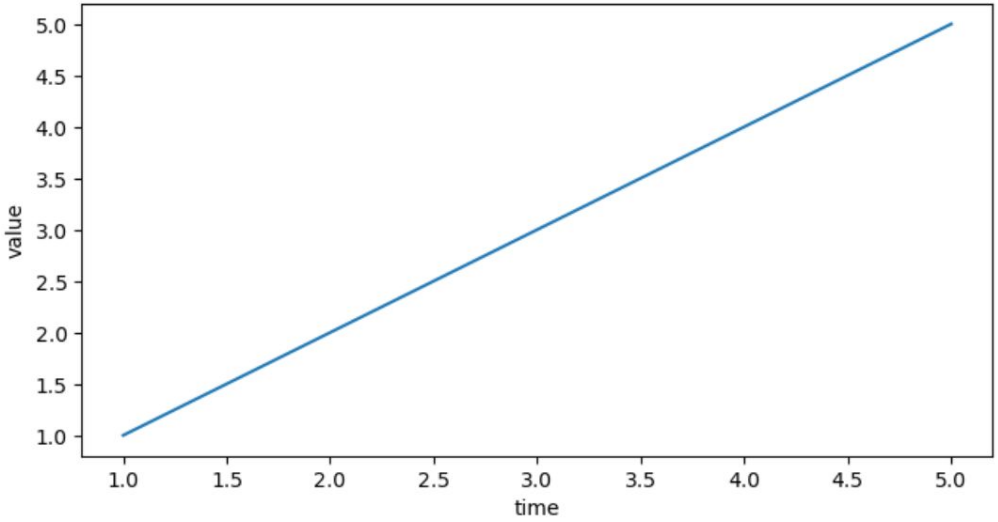


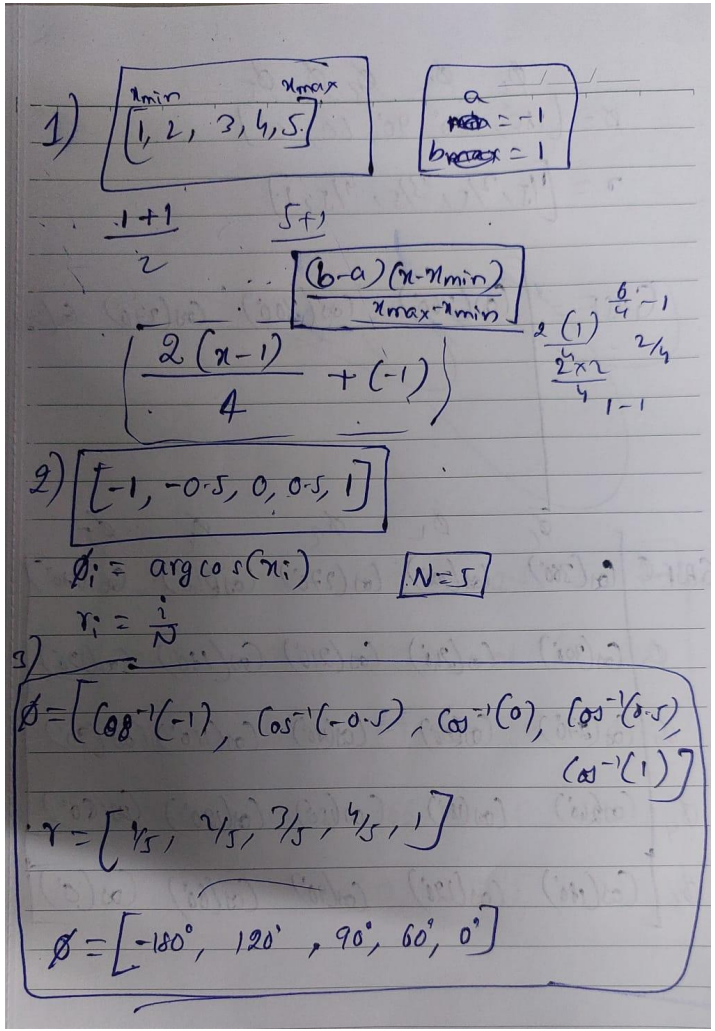
$t_1=\{x_1, x_2, \dots, x_n\}$



$t_2=\{-x_1, -x_2, \dots, -x_n\}$

Verification

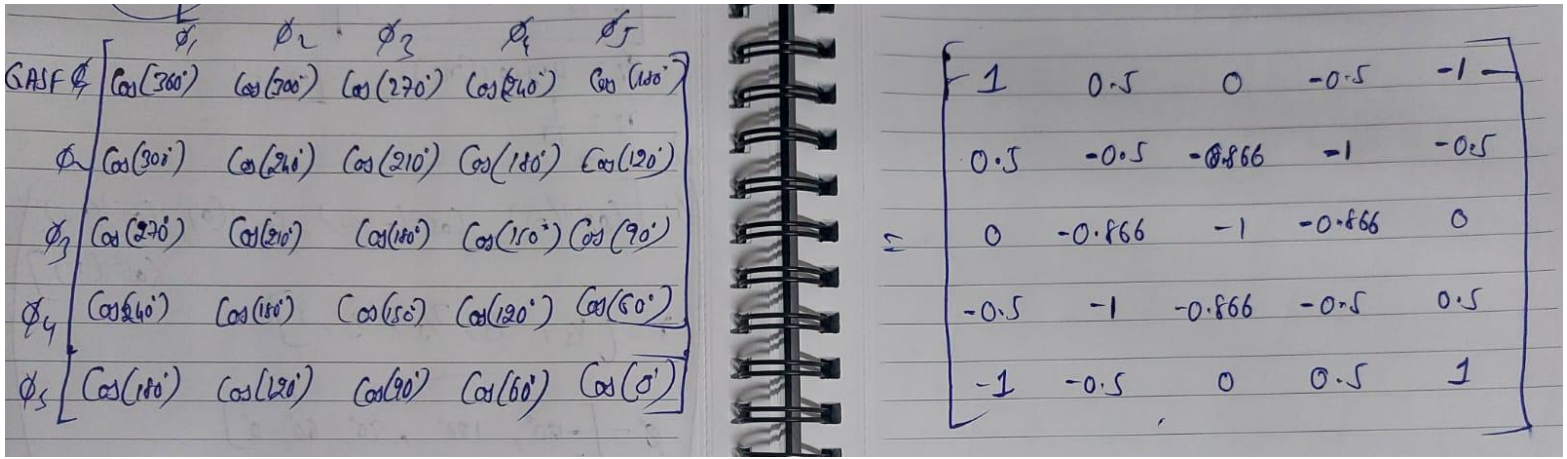




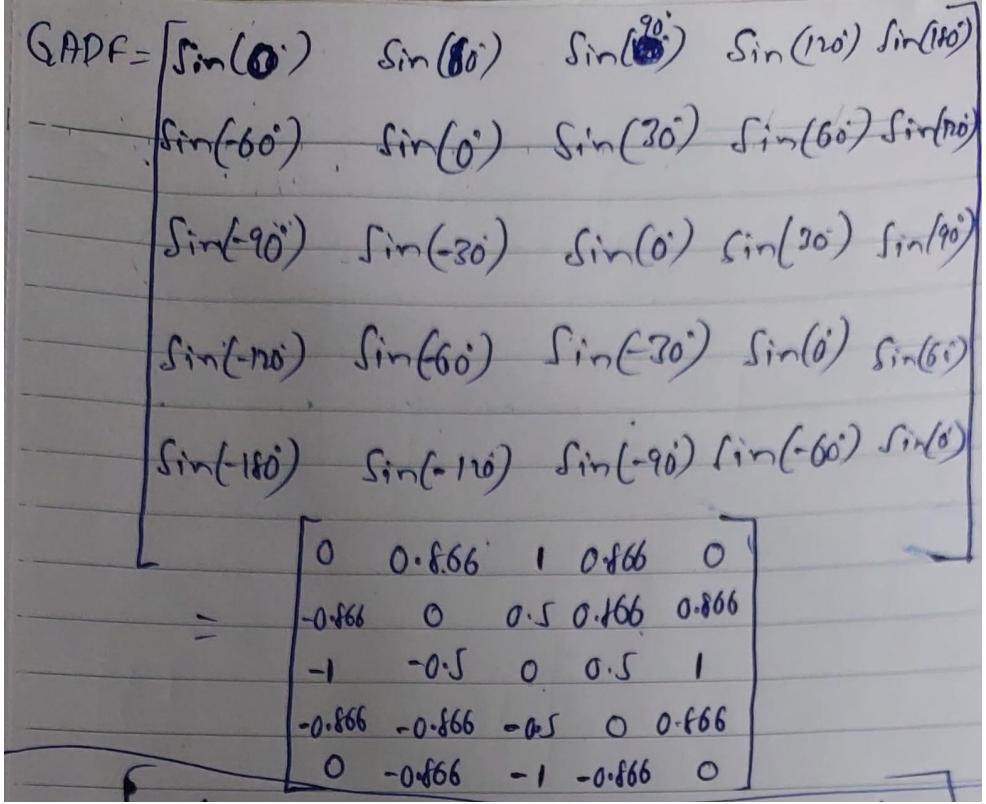
Verification GASF

```
X_gasf[0]
```

```
array([[ 1.        ,  0.5       , -0.        , -0.5       , -1.        ],
       [ 0.5       , -0.5       , -0.8660254, -1.        , -0.5       ],
       [-0.        , -0.8660254, -1.        , -0.8660254,  0.        ],
       [-0.5       , -1.        , -0.8660254, -0.5       ,  0.5       ],
       [-1.        , -0.5       ,  0.        ,  0.5       ,  1.        ]])
```



Verification GADF

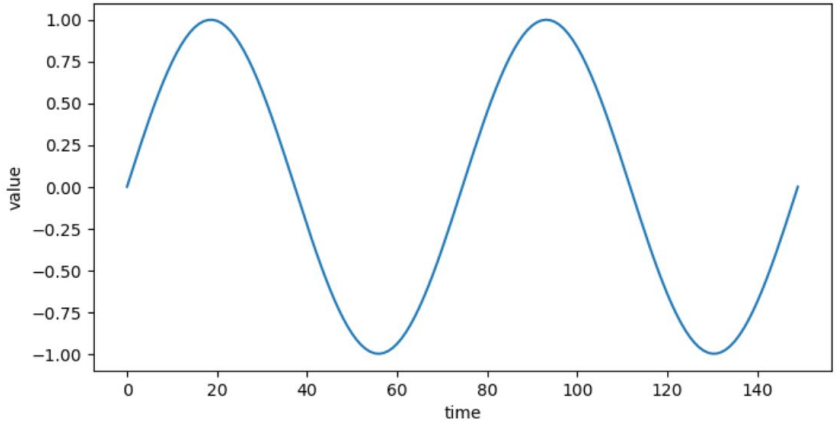


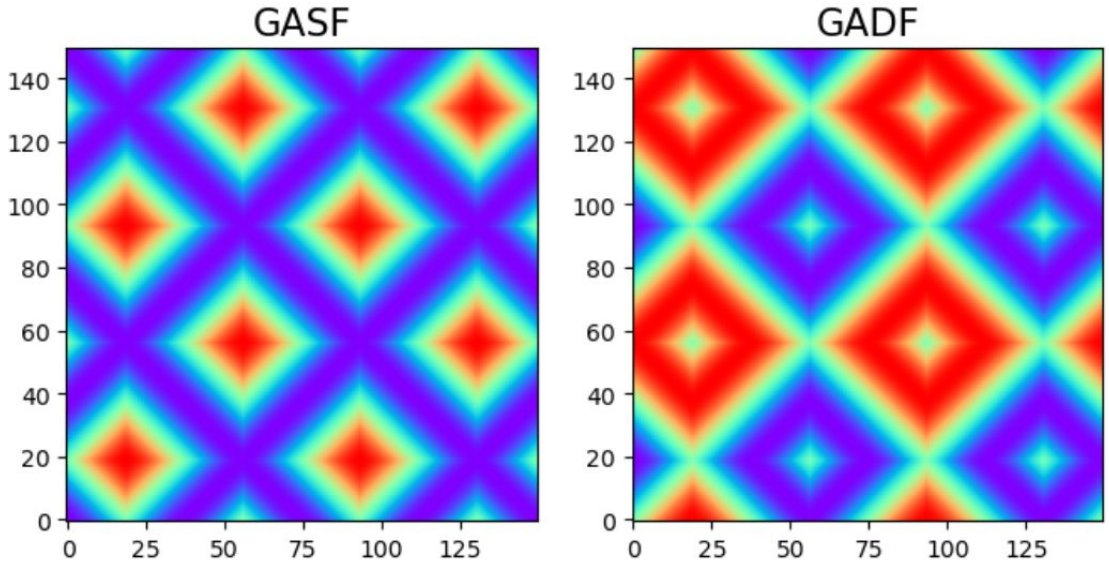

```
array([[ 0.        ,  0.8660254,  1.        ,  0.8660254,  0.        ],
       [-0.8660254,  0.        ,  0.5       ,  0.8660254,  0.8660254],
       [-1.        , -0.5       ,  0.        ,  0.5       ,  1.        ],
       [-0.8660254, -0.8660254, -0.5       ,  0.        ,  0.8660254],
       [-0.        , -0.8660254, -1.        , -0.8660254,  0.        ]])
```

Sine wave with and without noise

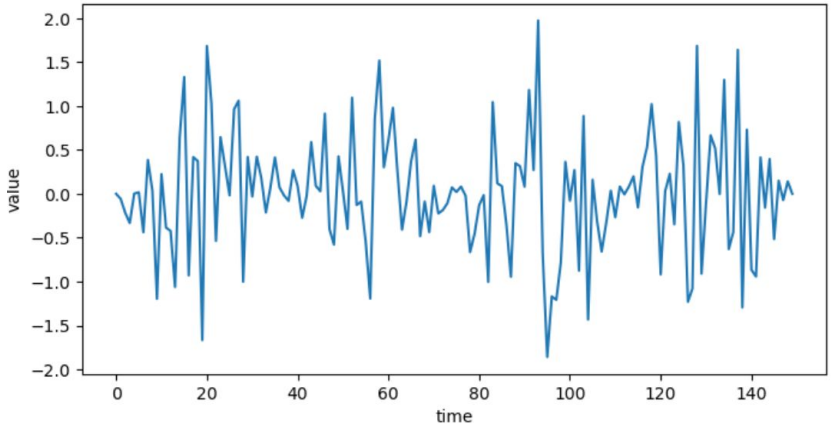
- time_points = np.linspace(0, 4 * np.pi, 150)
- y = np.sin(time_points)
- noise = np.random.normal(0,1,150)
- y_noise = [i*j for i, j in zip(y, noise)]

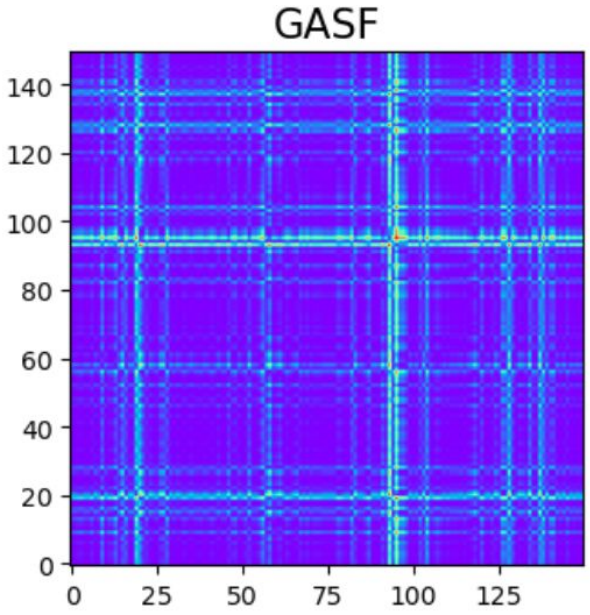
Without noise

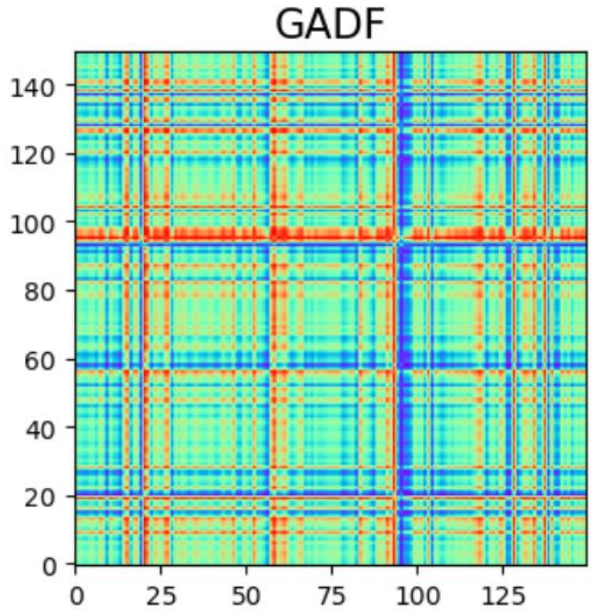




With noise


value
time
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
20
40
60
80
100
120
140


GASF
140
120
100
80
60
40
20
0
0
25
50
75
100
125


GADF
140
120
100
80
60
40
20
0
0
25
50
75
100
125

Student Confusion detection using EEG data

Introduction

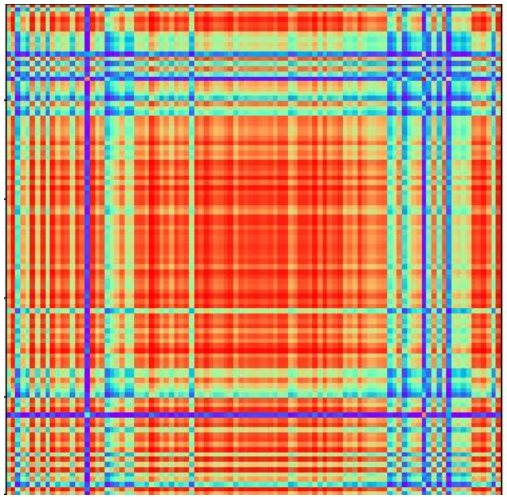
- The dataset contains data collected from 10 students while they were watching 10 videos each.
- Each video is about 2 min long and the data is collected at an interval of 0.5 seconds, so each time series has around 120 points.
- I converted the 1D time series data to an image using ‘Gramian angular field’ so that each (student, video) pair data can represent as an image.
- The dataset with images will now contain 100 rows.
- Now I used the image and its target label(0 or 1) for the binary classification task.

Dataset

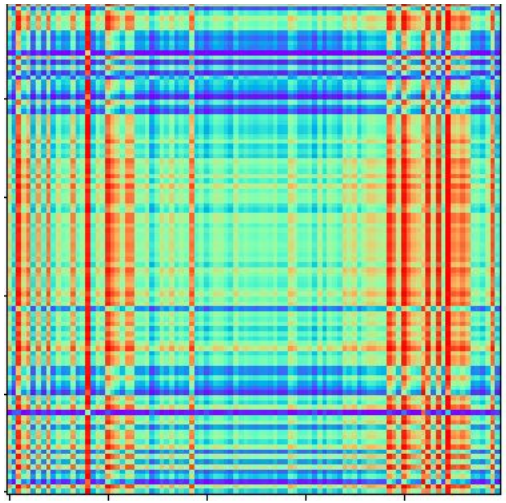
	SubjectID	VideoID	Attention	Mediation	Raw	Delta	Theta	Alpha1	Alpha2	Beta1	Beta2	Gamma1	Gamma2	predefinedlabel	user-definedlabeln	age	ethnicity	gender
0	0.0	0.0	56.0	43.0	278.0	301963.0	90612.0	33735.0	23991.0	27946.0	45097.0	33228.0	8293.0	0.0	0.0	25	Han Chinese	M
1	0.0	0.0	40.0	35.0	-50.0	73787.0	28083.0	1439.0	2240.0	2746.0	3687.0	5293.0	2740.0	0.0	0.0	25	Han Chinese	M
2	0.0	0.0	47.0	48.0	101.0	758353.0	383745.0	201999.0	62107.0	36293.0	130536.0	57243.0	25354.0	0.0	0.0	25	Han Chinese	M
3	0.0	0.0	47.0	57.0	-5.0	2012240.0	129350.0	61236.0	17084.0	11488.0	62462.0	49960.0	33932.0	0.0	0.0	25	Han Chinese	M
4	0.0	0.0	44.0	53.0	-8.0	1005145.0	354328.0	37102.0	88881.0	45307.0	99603.0	44790.0	29749.0	0.0	0.0	25	Han Chinese	M
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12806	9.0	9.0	64.0	38.0	-39.0	127574.0	9951.0	709.0	21732.0	3872.0	39728.0	2598.0	960.0	1.0	0.0	24	Han Chinese	F
12807	9.0	9.0	61.0	35.0	-275.0	323061.0	797464.0	153171.0	145805.0	39829.0	571280.0	36574.0	10010.0	1.0	0.0	24	Han Chinese	F
12808	9.0	9.0	60.0	29.0	-426.0	680989.0	154296.0	40068.0	39122.0	10966.0	26975.0	20427.0	2024.0	1.0	0.0	24	Han Chinese	F
12809	9.0	9.0	60.0	29.0	-84.0	366269.0	27346.0	11444.0	9932.0	1939.0	3283.0	12323.0	1764.0	1.0	0.0	24	Han Chinese	F
12810	9.0	9.0	64.0	29.0	-49.0	1164555.0	1184366.0	50014.0	124208.0	10634.0	445383.0	22133.0	4482.0	1.0	0.0	24	Han Chinese	F

12811 rows × 18 columns

Images generated for subject-0 and video-0



GASF



GADF

Model

- Model used for this binary classification task is CNN

```
gasf_model = models.Sequential()

gasf_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(im_size, im_size, 1)))
gasf_model.add(MaxPooling2D(pool_size=(2, 2)))
gasf_model.add(Dropout(0.25))

gasf_model.add(Conv2D(64, (3,3), activation='relu'))
gasf_model.add(MaxPooling2D(pool_size=(2, 2)))
gasf_model.add(Dropout(0.2))

gasf_model.add(Flatten())

gasf_model.add(Dense(64, activation='relu'))
gasf_model.add(Dropout(0.5))
gasf_model.add(Dense(1, activation='sigmoid'))

gasf_model.summary()
```

Results

- The results were generated using the k-fold validation **(k=10)**

EEG wave	GASF accuracy	GADF accuracy
Alpha1	71.00000023841858	64.00000065565109
Alpha2	74.00000035762787	75.99999964237213
Beta1	71.00000083446503	68.00000071525574
Beta2	73.99999976158142	68.99999976158142
Gamma1	66.00000083446503	63.99999976158142
Gamma2	65.00000029802322	62.99999952316284
Delta	79.9999988079071	73.00000071525574
Theta	72.99999982118607	74.00000035762787

Results-contd

- The results were generated using the k-fold validation **(k=5)**

EEG wave	GASF accuracy	GADF accuracy
Alpha1	61.99999928474426	56.99999988079071
Alpha2	71.99999928474426	69.9999988079071
Beta1	64.9999988079071	62.00000047683716
Beta2	61.00000023841858	63.00000071525574
Gamma1	65.0	64.9999988079071
Gamma2	63.99999976158142	60.000001192092896
Delta	72.99999952316284	65.99999904632568
Theta	67.99999833106995	70.99999904632568

Thank You!