Imperial College London

Convolutional Loss Functions in Deep Learning

Ranran Tao

September 2021

Imperial College London

ACSE 2020

Overview

- Theory
- Objective
- Implementation
- Test
- Outcome
- Conclusion

Imperial College London

Theory

1D - Calculation

- Autoencoder: $Input * w = Output$
- Input: *1D vector a.* Output: *1D vector b*
- Minimise: $g = \frac{1}{2}||Aw - b||^2$
- Wiener filter: $w = (A^T A)^{-1} A^T b$
- Loss function: $f = \frac{1}{2}||Tw||^2$
- Toeplitz matrix A:

$$A = \begin{bmatrix} a_1 & 0 & \cdots & 0 & 0 \\ a_2 & a_1 & \cdots & \vdots & \vdots \\ a_3 & a_2 & \cdots & 0 & 0 \\ \vdots & a_3 & \cdots & a_1 & 0 \\ a_{m-1} & \vdots & \ddots & a_2 & a_1 \\ a_m & a_{m-1} & \ddots & \vdots & a_2 \\ 0 & a_m & \ddots & a_{m-2} & \vdots \\ 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ \vdots & \vdots & \vdots & a_m & a_{m-1} \\ 0 & 0 & \vdots & \cdots & a_m \end{bmatrix}$$

2D - Calculation

- Autoencoder: $Input * w = Output$
- Input: *2D matrix a.* Output: *1D vector b*
- Minimise: $g = \frac{1}{2}||Aw - b||^2$
- Wiener filter: $w = (A^T A)^{-1} A^T b$
- Loss function: $f = \frac{1}{2}||Tw||^2$
- Toeplitz matrix A:

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 & 0 \\ A_2 & A_1 & \cdots & \vdots & \vdots \\ A_3 & A_2 & \cdots & 0 & 0 \\ \vdots & A_3 & \cdots & A_1 & 0 \\ A_{m-1} & \vdots & \ddots & A_2 & A_1 \\ A_m & A_{m-1} & \ddots & \vdots & A_2 \\ 0 & A_m & \ddots & A_{m-2} & \vdots \\ 0 & 0 & \cdots & A_{m-1} & A_{m-2} \\ \vdots & \vdots & \vdots & A_m & A_{m-1} \\ 0 & 0 & \vdots & \cdots & A_m \end{bmatrix}$$

, A_i *are Toeplitz matrices of each row of the 2D matrix a*

Imperial College London

Objective

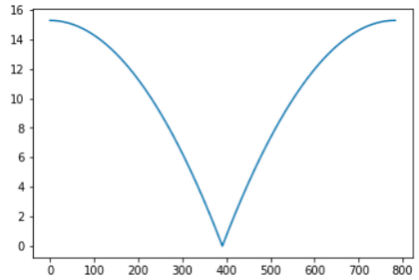
Optimisation of 1D and 2D Wiener filter calculation in deep networks.

- Time Efficiency Improvement
- Accuracy Improvement

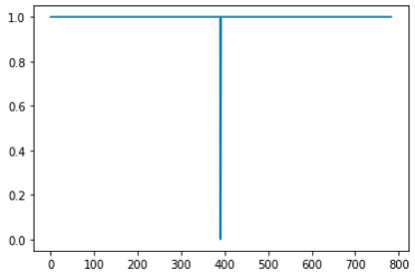
Implementation – 1D

Choose a penalty function

Penalty functions:

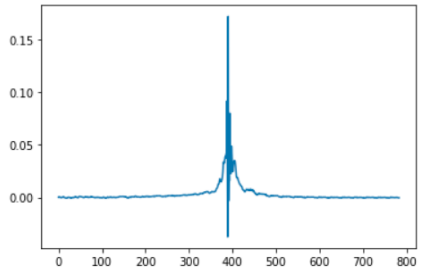


Smooth gaussian



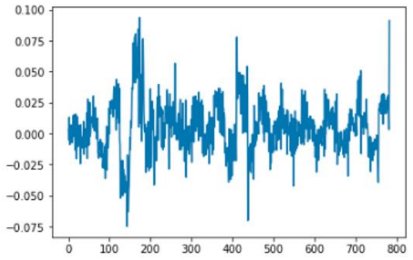
Sharp T

- Training - smooth gaussian as penalty function:

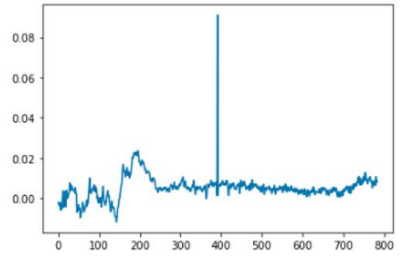



```
Epoch: 1000     Training Loss: 0.051588
torch.Size([1, 1, 28, 28])
<matplotlib.image.AxesImage at 0x7f487148a510>
```

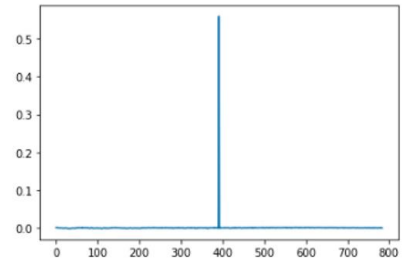
- Training - sharp T function as penalty function:




```
Epoch: 1        Training Loss: 0.190780
```




```
Epoch: 150      Training Loss: 0.019971
```




```
Epoch: 1000     Training Loss: 0.000237
```

Imperial College London

Implementation – 1D

Two different methods to generate the Toeplitz matrix

- First method:

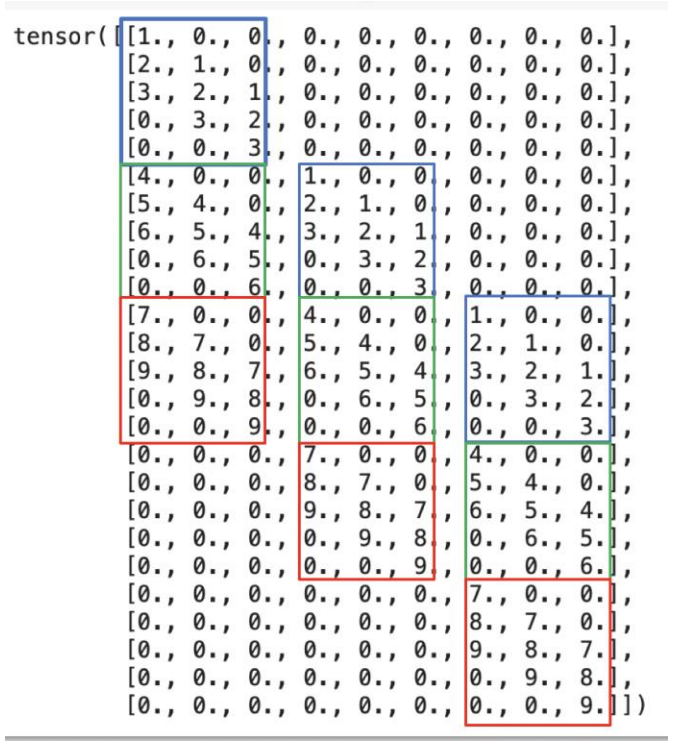
```
for i in range(h):
    A[i:i+h, i] = x[:]
```

- Second method:

```
for j in range(lw):
    A_T[j] = torch.roll(x, j)
```

Implementation – 2D

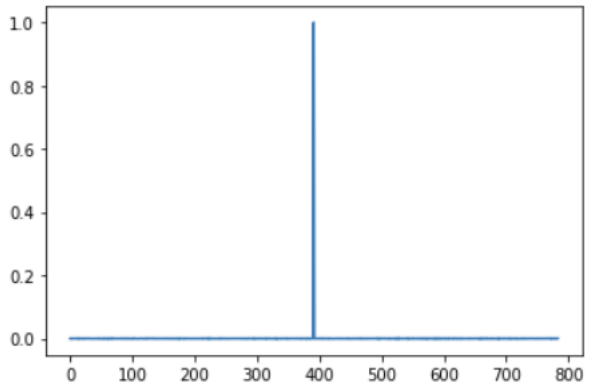
A sample doubly blocked Toeplitz matrix with a small size of 2D input matrix a:



, where 2D input matrix a = $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

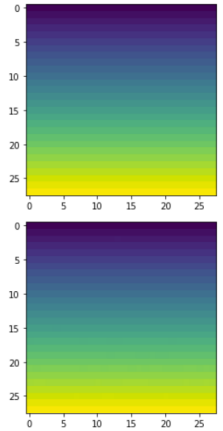
Test

- Check the shape of the filter:



Make the same input and output data, plot the filter. It shows that the shape of the filter is a delta function, which means the filter is correct.

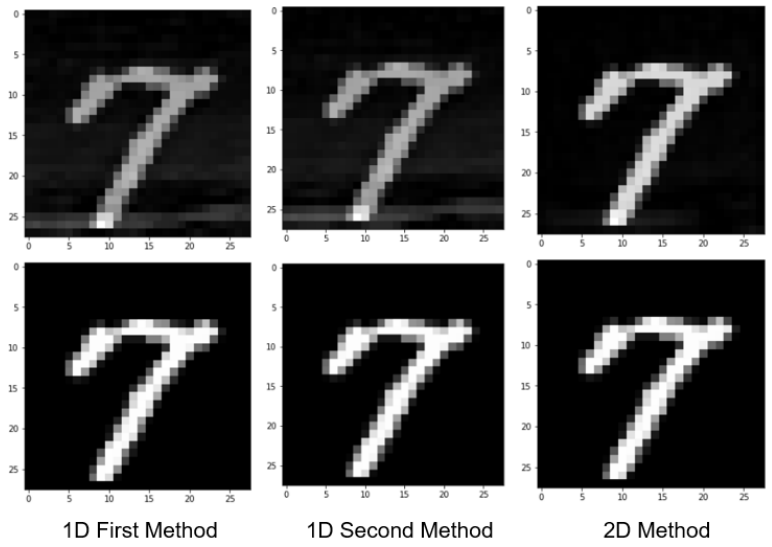
- Convolve the filter with a random image:



The output image is still the same with the input image after the convolution operation between the image and the filer. It means the filter is correct.

Outcome

Outcome Images:

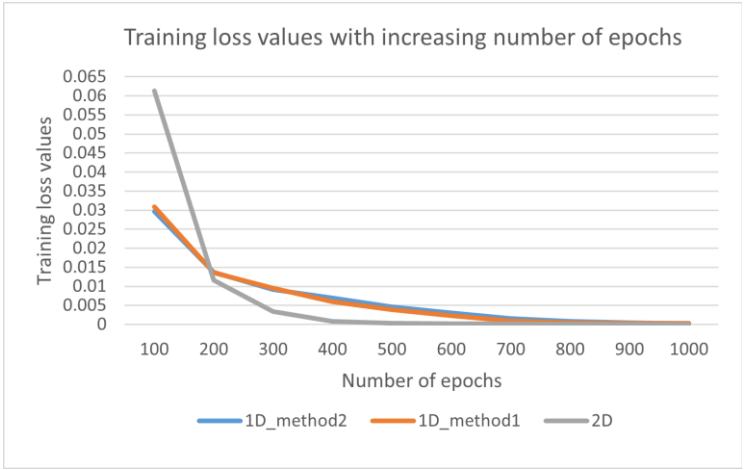


- Time efficiency:

Table2: Time Costs Comparison (1000 epochs)

1D First Method	1D Second Method	2D Method
33 seconds	28 seconds	29 seconds

- Convergence rate:



Imperial College London ACSE 2020

Conclusion

Optimisation Strategy:

- Using torch.roll when building the Toeplitz to improve time efficiency.
- Using 2D input data to improve convergence rate of training loss, which optimises image accuracy.

Strength and Limitations:

- Provided an advanced optimisation strategy for adaptive loss functions with Wiener filters in deep learning.
- Focused on one image.

Future Work:

- Increasing the batch size.
- Trying a deeper network.

Imperial College London

ACSE 2020

Thank you!