test_losses

February 22, 2022

[ ]:
```
import sys
sys.path.append('..')

import torch
import torch.nn.functional as F
import numpy as np
from scipy import signal

%load_ext autoreload
%autoreload 2
from networks import *
from utils import *
from losses import *

import matplotlib.pyplot as plt
from math import ceil
```

[ ]:
```
def get_middle_arr(arr, length):
    len_arr = len(arr)
    start = int(len_arr/2) - int(length/2)
    return arr[start:start+length]


def get_middle_arr2d(img, shape):
    x_len, y_len = shape[0], shape[1]
    x_start = int(img.shape[0]/2) - int(x_len/2)
    y_start = int(img.shape[1]/2) - int(y_len/2)
    return img[x_start:x_start+x_len, y_start:y_start+y_len]


def circle_img(shape, lag=(0, 0), fill_val=0, circle_val=1., radius=1.):
    im = torch.zeros(shape) + fill_val

    xarr = torch.linspace(-shape[0], shape[0], shape[0])
    yarr = torch.linspace(-shape[1], shape[1], shape[1])
    xx, yy = torch.meshgrid(xarr, yarr)
```

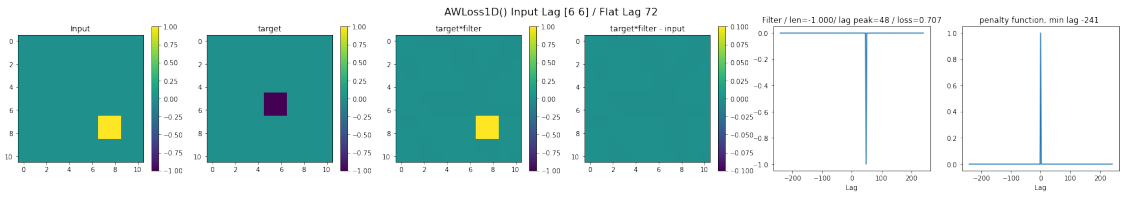
```
    lagy, lagx = lag
    dist = torch.sqrt((xx-lagx)**2 + (yy-lagy)**2)
    idx = torch.where(dist <= radius*2)
    im[idx[0], idx[1]] = circle_val

    return im

def square_img(shape, lag=(0, 0), fill_val=0, square_val=1., radius=1.):
    im = torch.zeros(shape) + fill_val

    xarr = torch.linspace(-shape[0], shape[0], shape[0])
    yarr = torch.linspace(-shape[1], shape[1], shape[1])

    lagx, lagy = lag

    idx = torch.where(xarr > 0 + lagx)[0][0]
    idy = torch.where(yarr > 0 + lagy)[0][0]
    im[idx-radius:idx+radius, idy-radius:idy+radius] = square_val

    return im
```

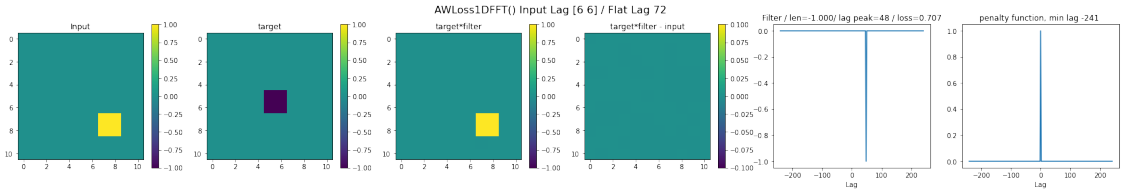
[ ]:
```
def test_loss(input, target, awi_loss, figtitle=None):
    assert input.shape == target.shape
    h, w = input.shape[2:]

    awi_loss.return_filters = True
    f = awi_loss(input, target)
    v, T = awi_loss.v_all, awi_loss.T_arr
    nd = len(v.shape) - 1

    if nd == 1:
        D = signal.convolve(target.flatten(start_dim=0), v[0], mode="full")
        D = get_middle_arr(D, h*w).reshape(h, w)
    elif nd == 2:
        D = signal.convolve2d(target.squeeze(0).squeeze(0), v[0])
        D = get_middle_arr2d(D, (h, w))#.reshape(h, w)
    else:
        raise Exception(" Only supporting filters of dimensions 1 and 2, but␣
 ↪found %g dimensions"%nd)

    fig, ax = plt.subplots(1, 6, figsize=(30, 4))
    if figtitle is not None:
        fig.suptitle(figtitle, fontsize=16)

    im = ax[0].imshow(input.squeeze(0).squeeze(0), vmin=-1, vmax=1)
    fig.colorbar(im, ax=ax[0])
    ax[0].set_title("Input")
```

```
   im = ax[1].imshow(target.squeeze(0).squeeze(0), vmin=-1, vmax=1)
   fig.colorbar(im, ax=ax[1])
   ax[1].set_title("target")

   im = ax[2].imshow(D, vmin=-1, vmax=1)
   fig.colorbar(im, ax=ax[2])
   ax[2].set_title("target*filter")

   im = ax[3].imshow(D - input.numpy().squeeze(0).squeeze(0), vmin=-input.
↪max()/10, vmax=input.max()/10)
   fig.colorbar(im, ax=ax[3])
   ax[3].set_title("target*filter - input")

   if nd == 1:
       xarr = torch.linspace(-len(v[0].flatten()), len(v[0].flatten()), ␣
↪len(v[0].flatten())).float()

       ax[4].plot(xarr, v[0].flatten())
       peak = int(xarr[torch.argmax(torch.abs(v[0].flatten())).item()])
       ax[4].set_title("Filter / len=%.3f/ lag peak=%g / loss=%.3f"%(v[0].
↪sum(), peak, f))
       ax[4].set_xlabel("Lag")

       ax[5].plot(xarr, T.detach())
       peak = int(xarr[torch.argmin(T).item()])
       ax[5].set_title("penalty function, min lag %g "%peak)
       ax[5].set_xlabel("Lag")

   elif nd == 2:
       xarr = np.linspace(-v[0].shape[0], v[0].shape[0], v[0].shape[0])
       yarr = np.linspace(-v[0].shape[1], v[0].shape[1], v[0].shape[1])
       xx, yy = np.meshgrid(xarr, yarr)


       im  = ax[4].contourf(xx, yy, v[0], levels=500)
       fig.colorbar(im, ax=ax[4])
       peaky, peakx = torch.where(v[0] == v[0].max())
       peaky, peakx  = int(yarr[peaky.item()]), int(xarr[peakx.item()])
       ax[4].set_title("Filter 2D / len={:.3f}/ lag peak=({:d}, {:d})/ loss={:.
↪3f}".format(v[0].sum(), peaky, peakx, f))
       ax[4].set_xlabel("Lag X")
       ax[4].set_ylabel("Lag Y")


       im = ax[5].contourf( xx, yy, T.detach(), levels=500)
       peaky, peakx = torch.where(T == T.max())
```

```
        peaky, peakx  = int(yarr[peaky]), int(xarr[peakx])
        fig.colorbar(im, ax=ax[5])
        ax[5].set_title("penalty function, min lag  ({:d}, {:d}) ".
↪format(peaky, peakx))
        ax[5].set_xlabel("Lag X")
        ax[5].set_ylabel("Lag Y")

    plt.show()
```

[ ]:
```
n = 11
losses = [
    AWLoss1D(store_filters=True),
    AWLoss1DFFT(store_filters=True, epsilon=3e-15),
    # AWLoss2D(store_filters=True),
    ]

P = torch.zeros((n, n))
for i in range(P.size(0)):
    P[i, :] = torch.zeros(n) + i + 1
P = P.unsqueeze(0).unsqueeze(0)

for awi_loss in losses:
    r = 1
    lag = np.array([6, 6])
    flatlag = n*lag[0] + lag[1]
    input = square_img((n, n), lag=lag, radius=r).unsqueeze(0).unsqueeze(0)
    target = square_img((n, n), lag=(0, 0), radius=r).unsqueeze(0).
↪unsqueeze(0)*-1
    test_loss(input, target, awi_loss, figtitle="{} Input Lag {} / Flat Lag {}".
↪format(awi_loss, lag, flatlag))
```

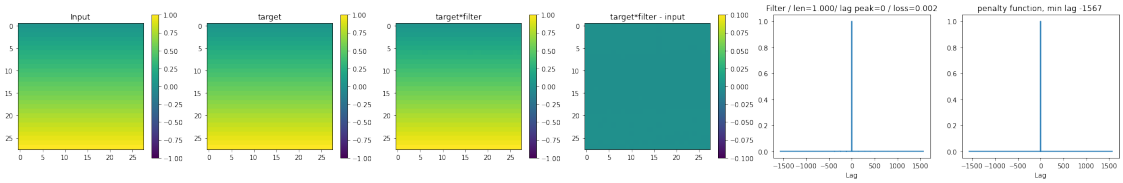


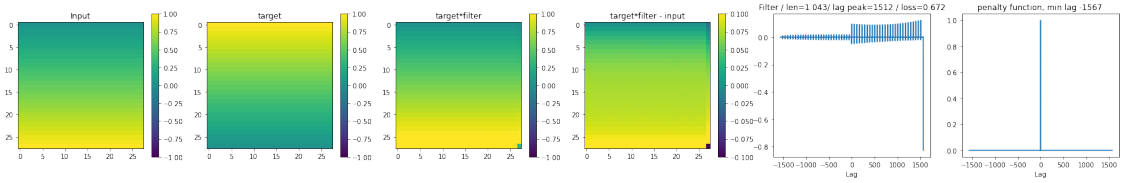


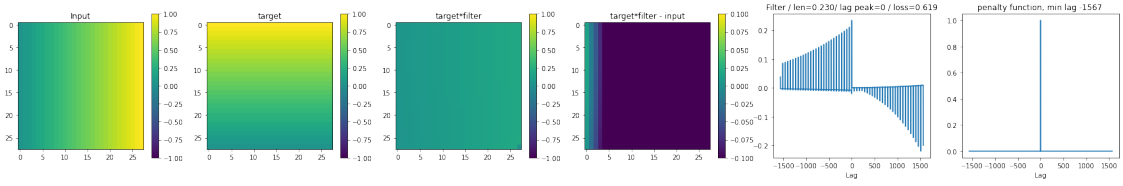
[ ]:
```
n = 28
e = 3e-15
losses = [AWLoss1D(store_filters=True, epsilon=e),
          AWLoss1DFFT(store_filters=True, epsilon=e),
        #   AWLoss2D(store_filters=True, epsilon=e)
        ]

for awi_loss in losses:
    print(awi_loss)


    P = torch.zeros(1,1,n,n)
    for i in range(n):
        P[0, 0, i, :] = (torch.ones(1, n) + i)/n

    input,  target = P[:], P[:]
    test_loss(input, target, awi_loss)

    input, target = P[:], torch.flipud(P[0, 0]).unsqueeze(0).unsqueeze(0)
    test_loss(input, target, awi_loss)

    input, target = torch.transpose(P, dim0=2, dim1=3), torch.flipud(P[0, 0]).
↪unsqueeze(0).unsqueeze(0)
    test_loss(input, target, awi_loss)

    input, target = torch.transpose(P, dim0=2, dim1=3), torch.flipud(P[0, 0]).
↪unsqueeze(0).unsqueeze(0)
    test_loss(input, target, awi_loss)

    for i in range(n):
        P[0, 0, :, i] = torch.ones(1, n) + i
    input, target = P[:], torch.fliplr(P[0, 0]).unsqueeze(0).unsqueeze(0)
    test_loss(input, target, awi_loss)

    circle = circle_img((n, n), lag=(0, 0), radius=8).unsqueeze(0).unsqueeze(0)
    target, input = circle, circle
    test_loss(input, target, awi_loss)

    circle = circle_img((n, n), lag=(15, 15), radius=8).unsqueeze(0).
↪unsqueeze(0)
    target, input = circle, circle
    test_loss(input, target, awi_loss)

    circle = circle_img((n, n), lag=(-15, -15), radius=8).unsqueeze(0).
↪unsqueeze(0)
    target, input = circle, circle
    test_loss(input, target, awi_loss)
```

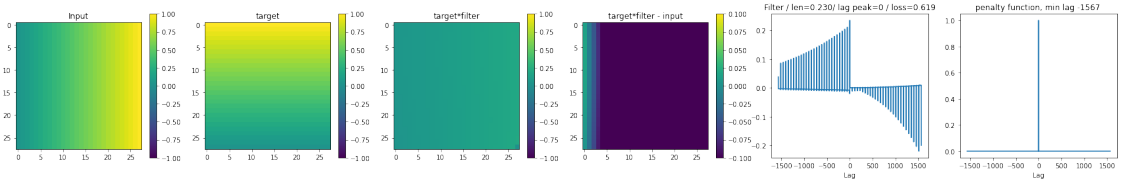
```
    circle = circle_img((n, n), lag=(0, -15), radius=8).unsqueeze(0).
↪unsqueeze(0)
    target, input = circle, circle
    test_loss(input, target, awi_loss)

    circle = circle_img((n, n), lag=(20, 20), radius=5).unsqueeze(0).
↪unsqueeze(0)
    input, target = circle, circle_img((n, n), lag=(0, 0), radius=5).
↪unsqueeze(0).unsqueeze(0)
    test_loss(input, target, awi_loss)

    lag = (0, 0)
    r = 2
    input = circle_img((n, n), lag=lag, radius=r).unsqueeze(0).unsqueeze(0)
    target = circle_img((n, n), lag=(0, 0), radius=r).unsqueeze(0).unsqueeze(0)
    test_loss(input, target, awi_loss)
```

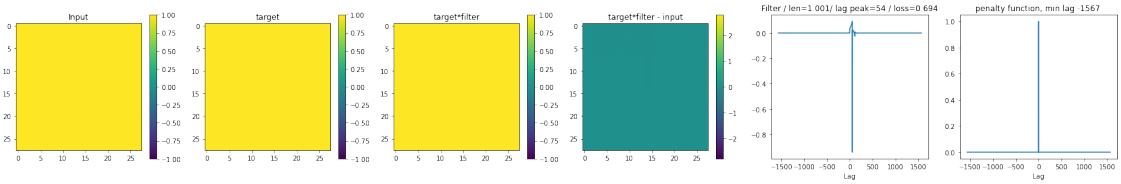
AWLoss1D()

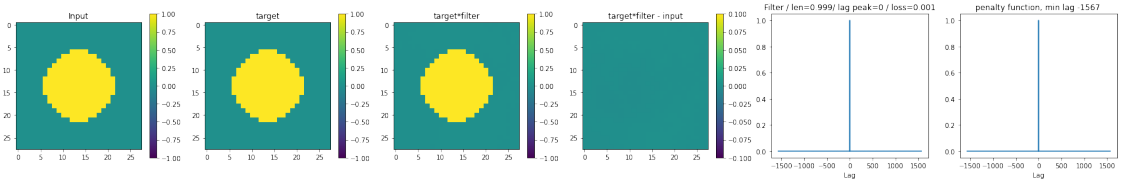


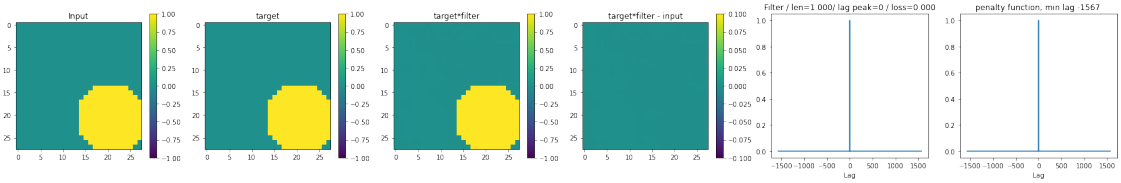


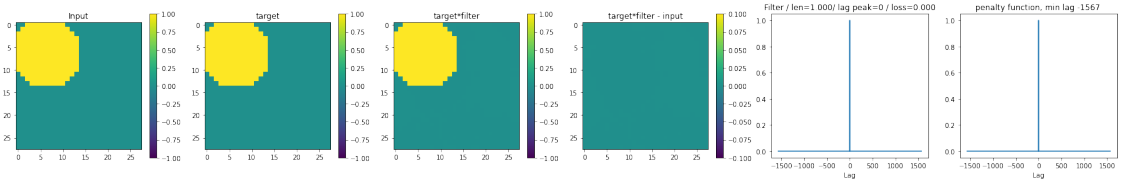


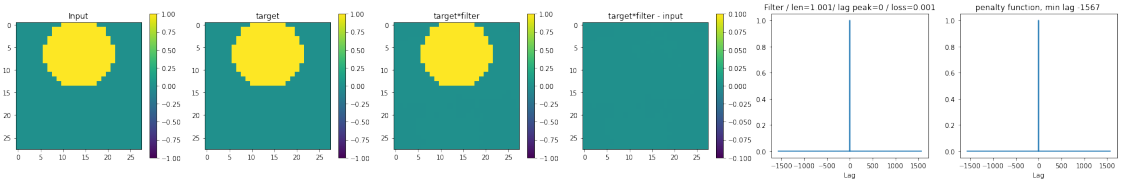

Input
target
target*filter
target*filter - input
Filter / len=0.230/ lag peak=0 / loss=0.619
penalty function, min lag -1567
Lag

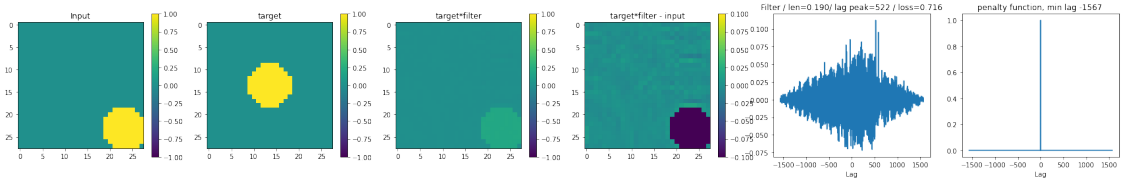

Input
target
target*filter
target*filter - input
Filter / len=1.001/ lag peak=54 / loss=0.694
penalty function, min lag -1567
Lag

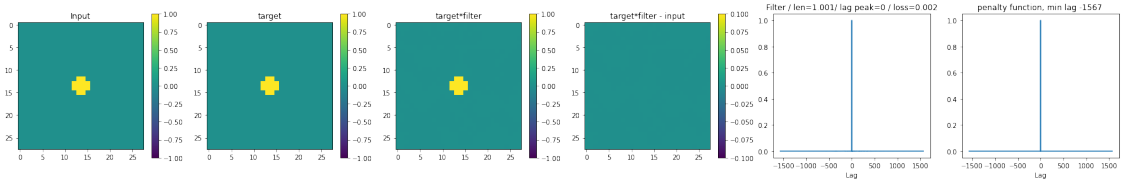

Input
target
target*filter
target*filter - input
Filter / len=0.999/ lag peak=0 / loss=0.001
penalty function, min lag -1567
Lag


Input
target
target*filter
target*filter - input
Filter / len=1.000/ lag peak=0 / loss=0.000
penalty function, min lag -1567
Lag

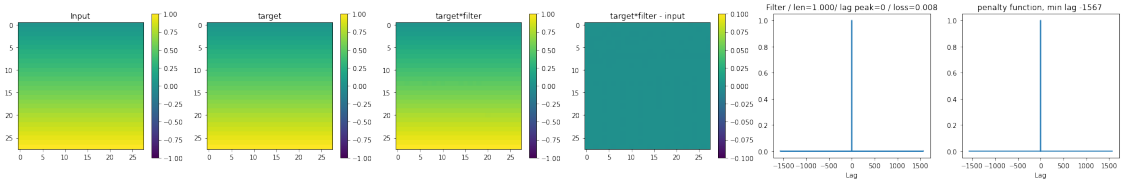

Input
target
target*filter
target*filter - input
Filter / len=1.000/ lag peak=0 / loss=0.000
penalty function, min lag -1567
Lag

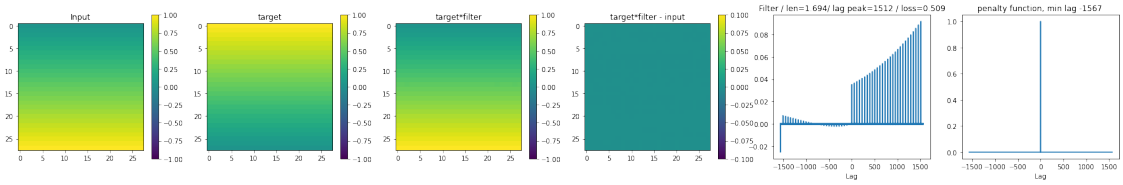

Input
target
target*filter
target*filter - input
Filter / len=1.001/ lag peak=0 / loss=0.001
penalty function, min lag -1567
Lag

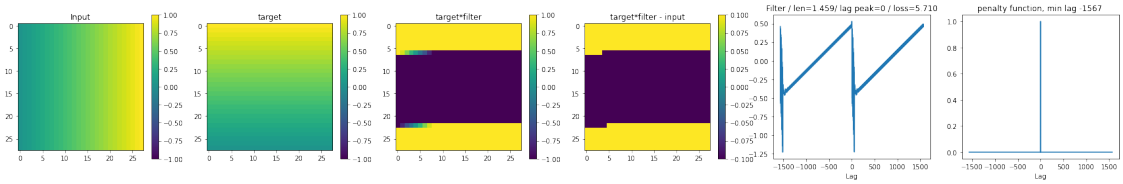


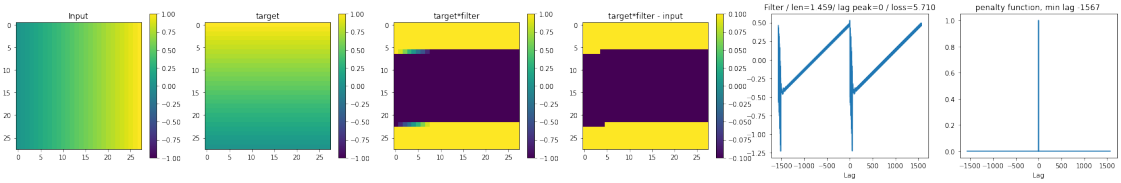


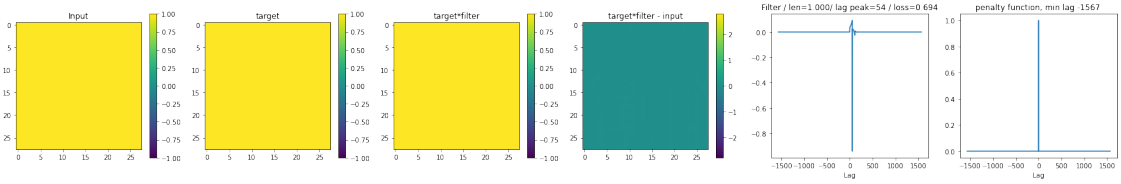
`AWLoss1DFFT()`

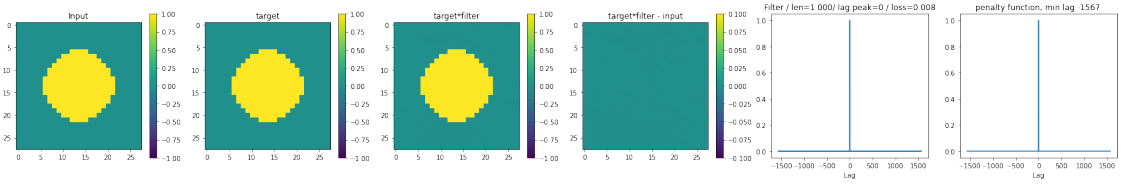


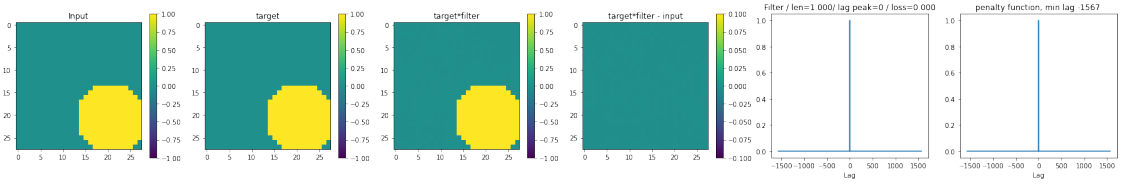


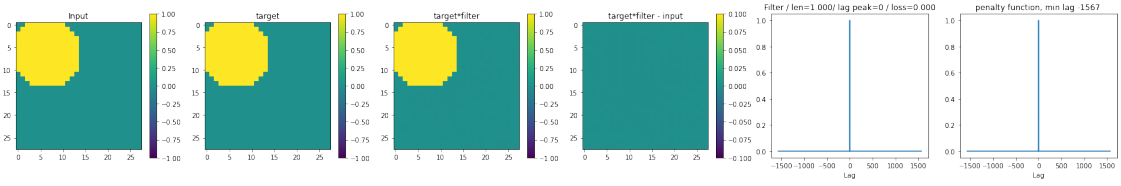


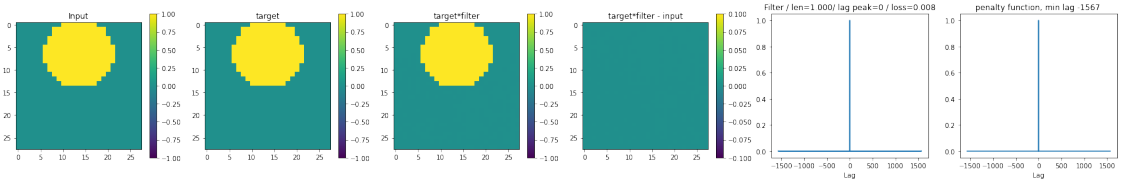

Input
target
target*filter
target*filter - input
Filter / len=1.459/ lag peak=0 / loss=5.710
penalty function, min lag: 1567
Lag

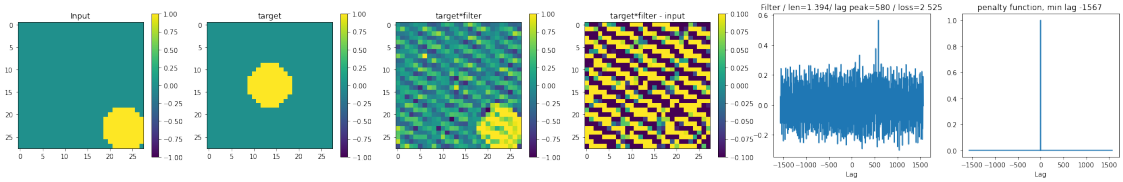

Input
target
target*filter
target*filter - input
Filter / len=1.000/ lag peak=54 / loss=0.694
penalty function, min lag: 1567
Lag

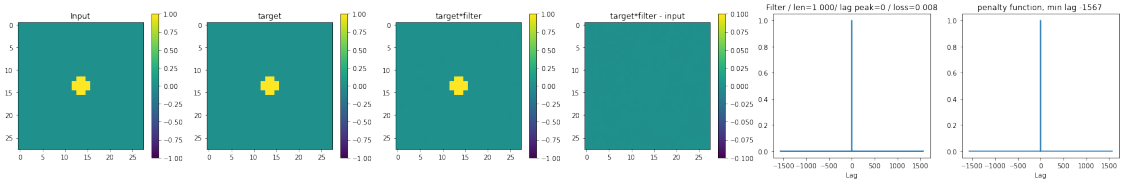

Input
target
target*filter
target*filter - input
Filter / len=1.000/ lag peak=0 / loss=0.008
penalty function, min lag: 1567
Lag


Input
target
target*filter
target*filter - input
Filter / len=1.000/ lag peak=0 / loss=0.000
penalty function, min lag: 1567
Lag


Input
target
target*filter
target*filter - input
Filter / len=1.000/ lag peak=0 / loss=0.000
penalty function, min lag: 1567
Lag


Input
target
target*filter
target*filter - input
Filter / len=1.000/ lag peak=0 / loss=0.008
penalty function, min lag: 1567
Lag


Input
target
target*filter
target*filter - input
Filter / len=1.394/ lag peak=580 / loss=2.525
penalty function, min lag :1567
Lag


Input
target
target*filter
target*filter - input
Filter / len=1.000/ lag peak=0 / loss=0.008
penalty function, min lag :1567
Lag

[ ]: