Pixel explanations for a deep neural network diabetic retinopathy classification model

Let's import some libraries required for the notebook to work:

In [1]:

```
import random
import os

import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision.transforms as transforms
import torchvision.datasets as datasets

import numpy as np

import quadratic_weighted_kappa as qwk
import transforms_extra
import models

import matplotlib
import matplotlib.pyplot as plt

import math

%matplotlib inline

import visualization_rev1 as vis
```

Let's define some constants related with the model:

In [2]:

```
workers = 4
batch_size = 1
mean = [0.383, 0.266, 0.190]
std = [0.291, 0.211, 0.171]
normvect = [-mean[0]/std[0], -mean[1]/std[1], -mean[2]/std[2]]
input_size = 640
input_channels = 3
# number of rotated configurations
configurations = 5
hidden_features = 64
```

And now, let's load the test dataset:

In [3]:

```
# Data loading code
testdir = '../input/640_itaka'
normalize = transforms.Normalize(mean=mean, std=std)
dataset_test = datasets.ImageFolder(testdir, transforms.Compose([
    transforms.Scale(input_size),
    transforms.ToTensor(),
    normalize,
    transforms_extra.CenteredCircularMaskTensor(input_size, normvect),
]))
```

```
/home/jordi/anaconda3/lib/python3.6/site-packages/torchvision-0.2.0-py3.6.eg
g/torchvision/transforms/transforms.py:156: UserWarning: The use of the tran
sforms.Scale transform is deprecated, please use transforms.Resize instead.
```

Let's load the previously trained model:

In [4]:

```
# load the model
resume = 'models/ret6_bn/model_best-QWKval0814.pth.tar'
print("=> loading checkpoint '{}'".format(resume))
checkpoint = torch.load(resume)
start_epoch = checkpoint['epoch']
best_prec1 = checkpoint['best_prec1']
import models
model = models.ret6_bn()
model.load_state_dict(checkpoint['state_dict'])
print("=> loaded checkpoint '{}' (epoch {})".format(resume, checkpoint['epoch']))
```

```
=> loading checkpoint 'models/ret6_bn/model_best-QWKval0814.pth.tar'
=> loaded checkpoint 'models/ret6_bn/model_best-QWKval0814.pth.tar' (epoch 6
38)
```

We instantiate the previously defined model with the parameters of the pretrained one loaded before:

In [5]:

```
me = models.model_explainable(model)
#me = me.cuda()
me = me.cpu()
me.eval()
```

Out[5]:

```
model_explainable(
  (rf3): Sequential(
    (0): Conv2d (3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
  )
  (rf5): Sequential(
    (0): Conv2d (16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
  )
  (rf9): Sequential(
    (0): Conv2d (16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
```

Now, we load a image from the dataset. This image is the one that will be analyzed:

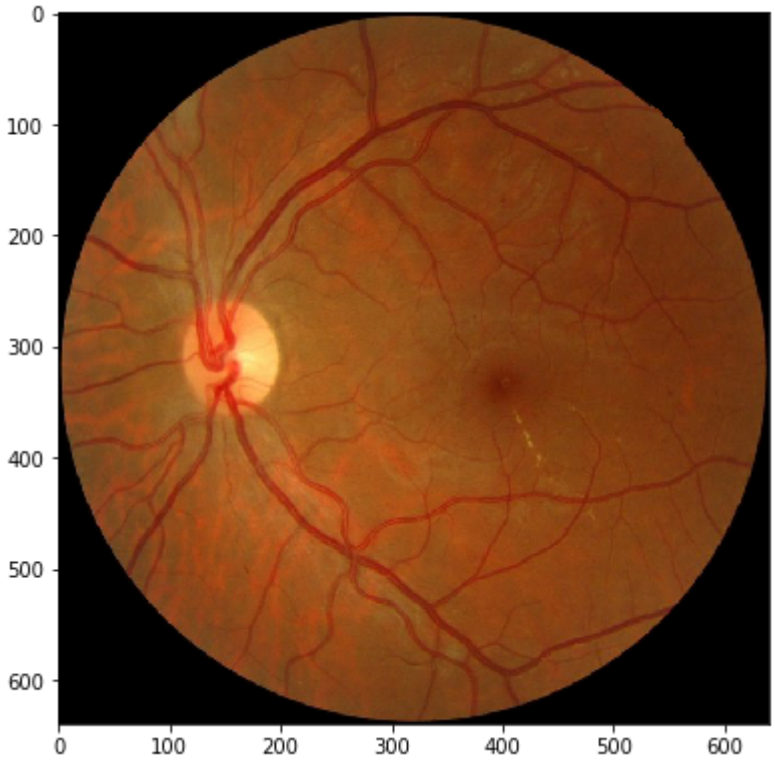
In [6]:

```
dataset = dataset_test

#image_nr = 9857
image_nr = 9

#image_nr = int(len(dataset.imgs)*random.random())

batch = torch.zeros(1, input_channels, input_size, input_size)
target = torch.zeros(1).long()

img, img_target = dataset.__getitem__(image_nr)
path, _ = dataset.imgs[image_nr]
file = os.path.basename(path)

batch[0].copy_(img)

if next(me.parameters()).is_cuda:
    batch = batch.cuda()

input = torch.autograd.Variable(batch, volatile=True)
target[0] = img_target
```

Let's visualize the image:

In [7]:

```
input_rgb = batch.clone()
#print(input_rgb.size())
for i in range(3):
    input_rgb[0][i].mul_(std[i])
    input_rgb[0][i].add_(mean[i])

input_rgb[torch.lt(input_rgb, 0)] = 0.0
input_rgb[torch.gt(input_rgb, 1)] = 1.0

vis.plot(input_rgb)

#del input_rgb
```



Let's calculate with the model the predicted classification class. We also calculate the intermediate values of the activations of every layer

In [8]:

```
#output, features, rf637, rf509_p, rf381, \
#rf253_p, rf189, rf125_p, rf93, rf61_p, rf45, \
#rf29_p, rf21, rf13_p, rf9, rf5_p, rf3
layer_vals = me.forward(input)
layer = 17
output = layer_vals[layer]

print("Image nr:" + str(image_nr))
print('File: ' + dataset.imgs[image_nr][0])
imagename = os.path.splitext(os.path.basename(dataset.imgs[image_nr][0]))[0]
print("Image name: " + imagename)
print("Tag")
print('Output before softmax: ')
print(output)
prob_pred_class, pred_class = output.max(1)
pred_class = pred_class[0]
print("Predicted class: " + str(pred_class))
print('Target: ')
print(target)
```

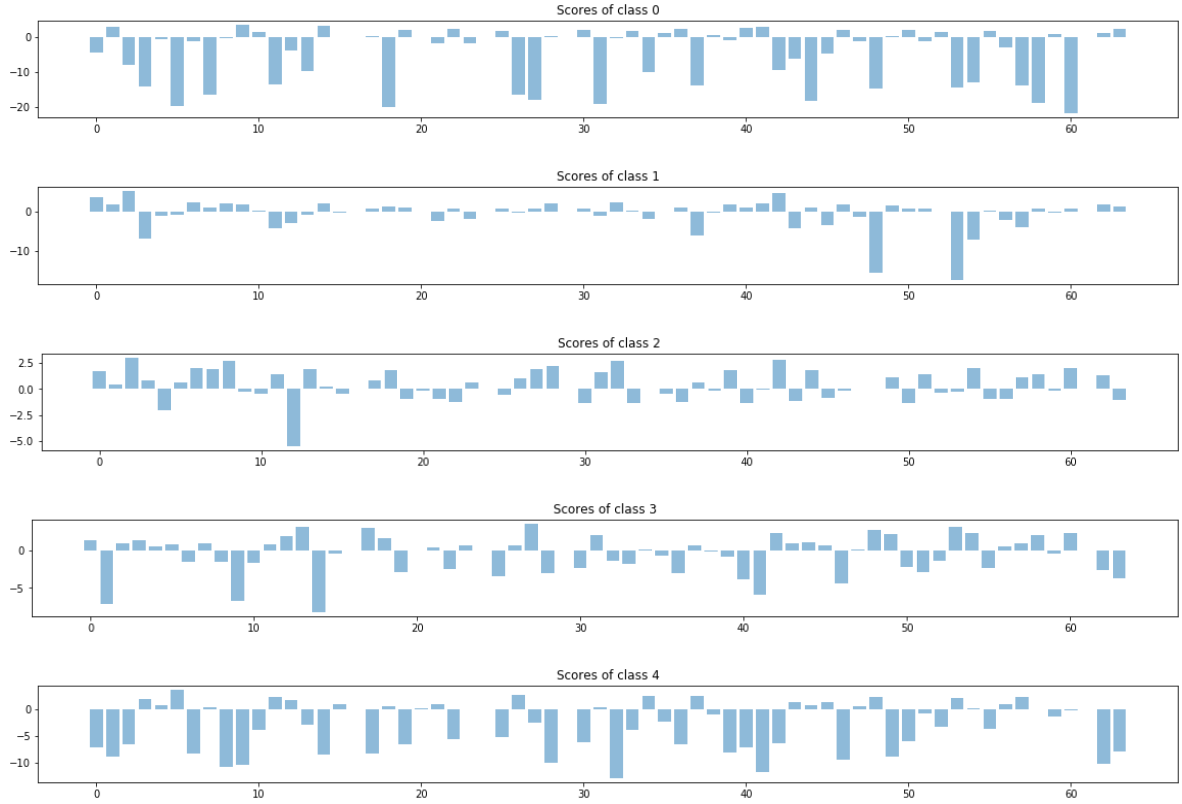
```
Image nr:9
File: ../input/640_itaka/itaka/0NGPP8431Y6KYVMJ_OS_ND.jpeg
Image name: 0NGPP8431Y6KYVMJ_OS_ND
Tag
Output before softmax:
Variable containing:
-295.0830  -30.9120   21.7021  -34.7909 -194.4052
[torch.FloatTensor of size 1x5]

Predicted class: Variable containing:
 2
[torch.LongTensor of size 1]

Target:

 0
[torch.LongTensor of size 1]
```

Last layer feature activations

And now we plot the feature activations in the last layer of the model previous to the output layer:

In [9]:

```
layer = 16
features = layer_vals[layer]
score_features = torch.mul(me.lc[0].weight, features.expand(5,64))
objects = [str(i) for i in range(64)]
y_pos = np.arange(len(objects))

for i in range(5):
    feat = score_features.data[i].tolist()
    fig = plt.figure(figsize=(20,10))
    fig.add_subplot(5, 1, i+1)
    plt.bar(y_pos, feat, align='center', alpha=0.5)
    plt.title('Scores of class {}'.format(i))
#plt.xticks(y_pos, objects)
#plt.ylabel('f')

plt.show()
```



We create a variable to sum the constants contributions of every layer mapped into the input space. Finally we will add this value to the values calculated for the input space

In [10]:

```
score_accumulated_k = torch.autograd.Variable(torch.zeros(5, 1, input_size, input_size), vo
```

Score propagation through average pooling layer:

In [11]:

```
layer = 15
rf637 = layer_vals[layer]
score_rf637 = me.lc[0].weight.unsqueeze(2).unsqueeze(3).expand(5,64,4,4) * rf637.expand(5,6
vis.stats(score_rf637)
```

```
Max: 0.7375839352607727, Min: -3.928438186645508, Avg: -0.10426702530657224,
Std:0.4164113856429738
torch.Size([5, 64, 4, 4])
```

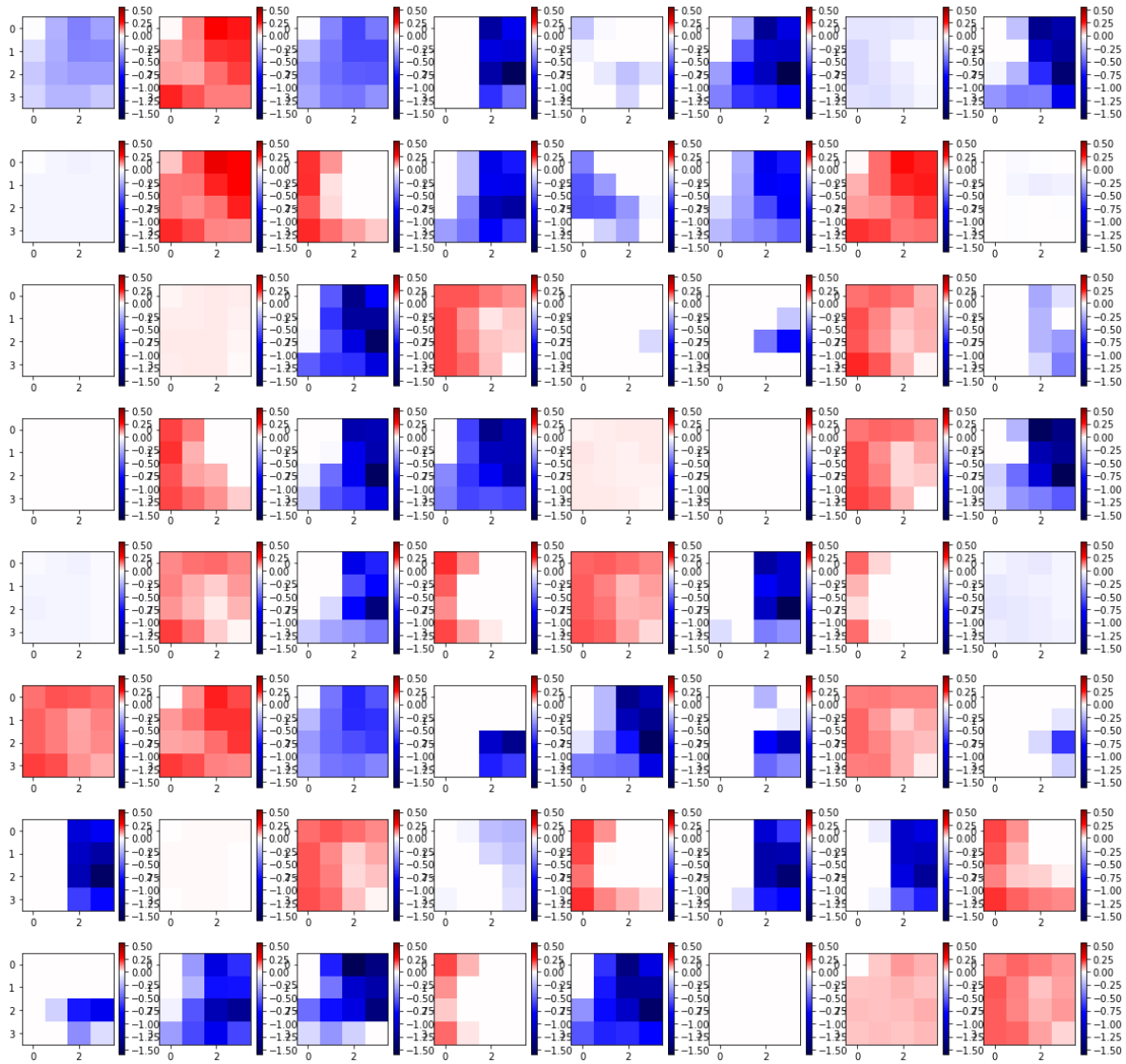
Out[11]:

```
(0.7375839352607727,
 -3.928438186645508,
 -0.10426702530657224,
 0.4164113856429738)
```

Scores of the individual feature maps of the 15th layer

In [12]:

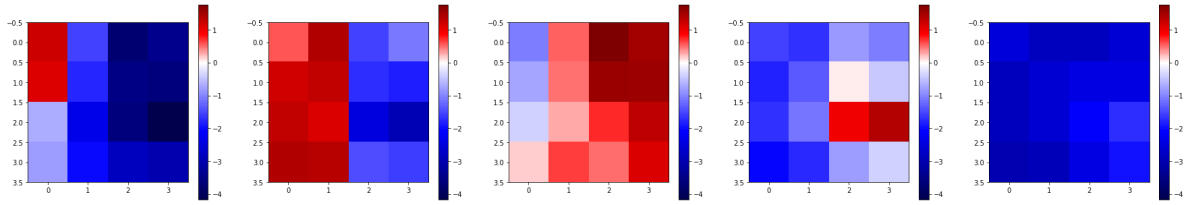
```
vis.plot_scores(img_target,score_rf637)
```



Aggregate feature maps scores of the 15th layer

In [13]:

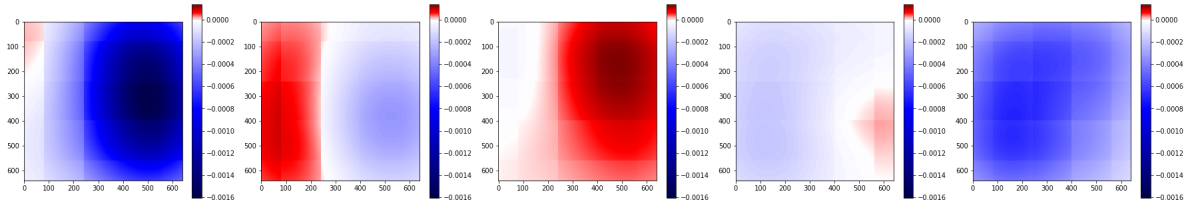
```
#vis.plot_scores(img_target,score_rf637.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf637.sum(1).unsqueeze(0), figsize=(50,50))
```



Aggregated scores of 15th layer mapped into pixel space using a 2d-normal 2std=rf prior

In [14]:

```
tmp = vis.map_scores_to_input_sz(score_rf637.sum(1).unsqueeze(1), 637)
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
#vis.plot_scores(img_target,tmp, figsize=(50,50))
print(tmp.view(-1).sum())
del tmp
```



```
Variable containing:
-533.8472
[torch.FloatTensor of size 1]
```

Score propagation through classification layer (2x2 convolution)

In [15]:

```
layer = 14
rf509_p = layer_vals[layer]
rf509, mp_idx = F.max_pool2d(rf509_p, 2, 2, return_indices = True)  # 1x64x5x5
score_rf509, score_k637 = vis.propagate_score_through_conv2d_sz2_pad0(score_rf637, rf509, r
#del val, rf509_p
vis.stats(score_rf509)
```

```
Max: 7.192414283752441, Min: -11.626453399658203, Avg: -0.09069047372631675,
Std:0.7450446762413168
torch.Size([5, 64, 5, 5])
```

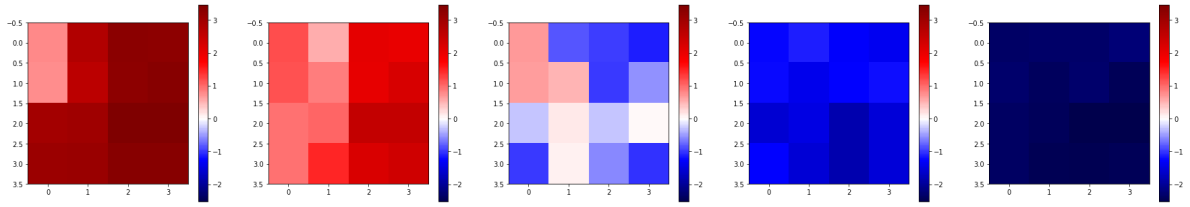
Out[15]:

```
(7.192414283752441,
 -11.626453399658203,
 -0.09069047372631675,
 0.7450446762413168)
```

In [16]:

```
# sum of constant scores of the layer
score_k637 = score_k637.sum(1).unsqueeze(1)
print(score_k637.size())
vis.plot_scores(0,score_k637.sum(1).unsqueeze(0), figsize=(50,50))
#vis.plot_scores(img_target,score_k637, figsize=(50,50))
```

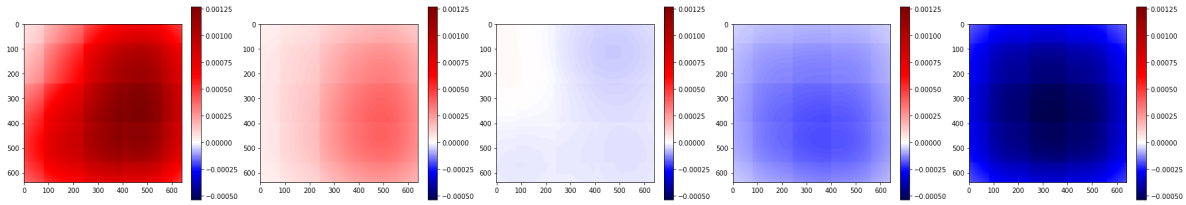
```
torch.Size([5, 1, 4, 4])
```



In the previous map, every pixel corresponds to a receptive field. There is superposition between the activations showed in the above map. We now from bibliography that the receptive fields have a nearly gaussian detection capability. Considering a 2 stdev gaussian model over the receptive field, we can map the activations to input space. The result is shown below:

In [17]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k637, 637)
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
print(score_k637.sum())
print(score_kmapped.sum())
#del score_k637
```

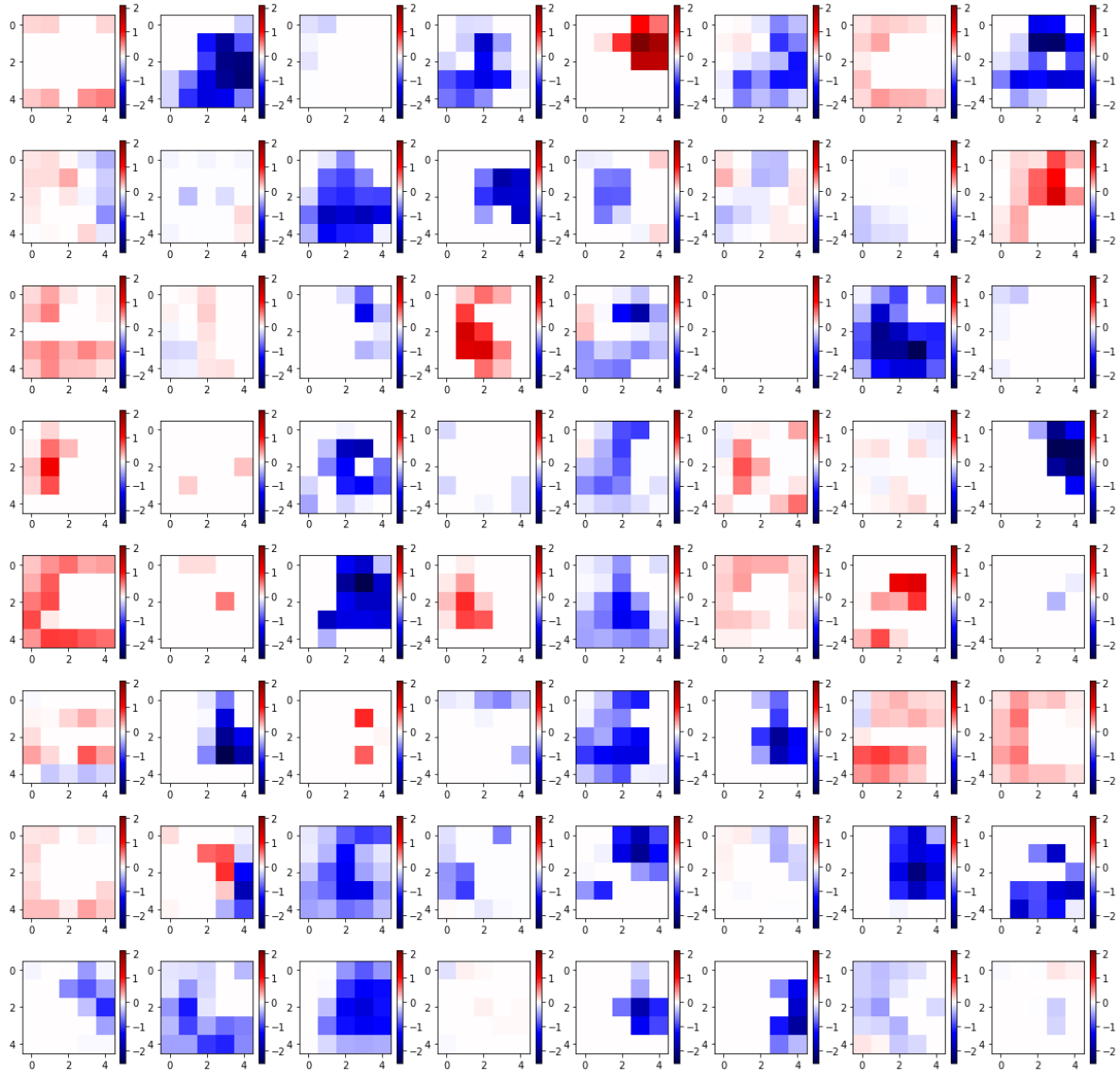


```
Max: 0.0012693775352090597, Min: -0.0005376714398153126, Avg: 9.359206684278
862e-05, Std:0.0004256944493931067
torch.Size([5, 1, 640, 640])
Variable containing:
 191.6765
[torch.FloatTensor of size 1]

Variable containing:
 191.6766
[torch.FloatTensor of size 1]
```

Scores of the feature maps of the 14th layer

In [18]:

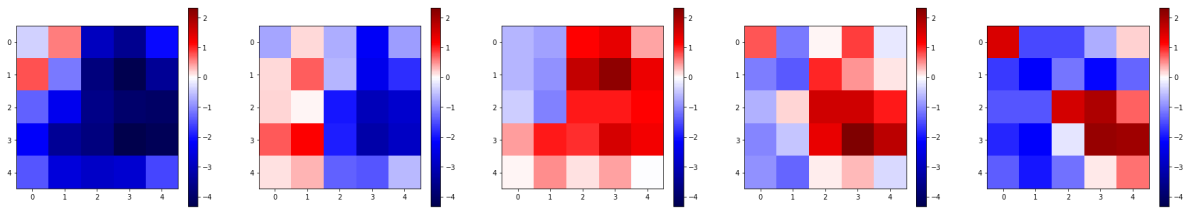
```
vis.plot_scores(img_target,score_rf509)
```



Aggregated feature map scores of the 14th layer

In [19]:

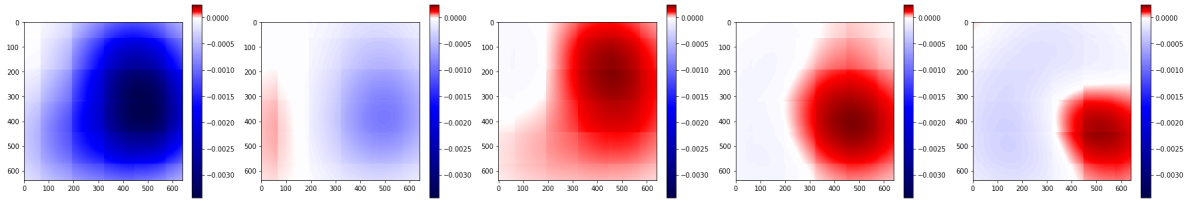
```
#vis.plot_scores(img_target,score_rf509.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf509.sum(1).unsqueeze(0), figsize=(50,50))
```



Aggregated feature map scores of the 14th layer mapped into input space

In [20]:

```
tmp = vis.map_scores_to_input_sz(score_rf509.sum(1).unsqueeze(1), 509)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [21]:

```
score_mp = vis.propagate_score_through_maxpool_sz2x2_st2x2(score_rf509, mp_idx)
#del score_rf509, mp_idx
layer = 13
rf381 = layer_vals[layer]
score_rf381, score_k509 = vis.propagate_score_through_conv2d_sz3_pad1(score_mp, rf381, rf50
#del score_mp, rf381
vis.stats(score_rf381)
vis.stats(score_k509)
```

```
Max: 8.579278945922852, Min: -13.765203475952148, Avg: -0.00916567132930409,
Std:0.3303136898627218
torch.Size([5, 64, 10, 10])
Max: 4.873049736022949, Min: -12.371910095214844, Avg: -0.01350694653382534
9, Std:0.34287225469358645
torch.Size([5, 64, 10, 10])
```

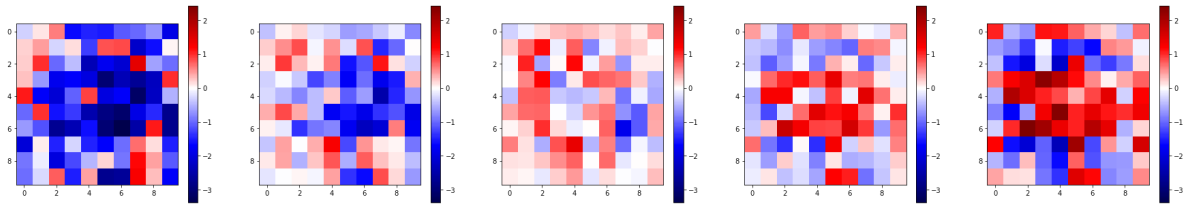
Out[21]:

```
(4.873049736022949,
 -12.371910095214844,
 -0.013506946533825349,
 0.34287225469358645)
```

In [22]:

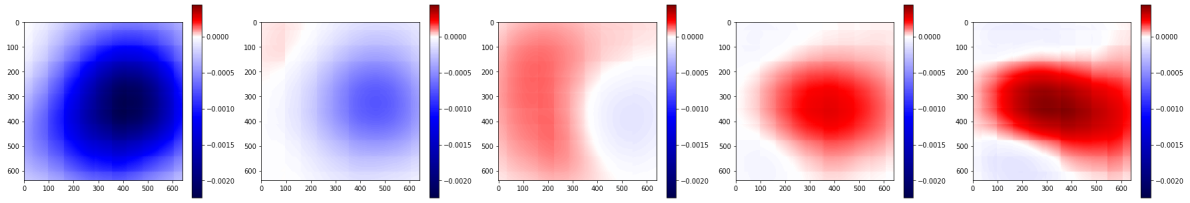
```
score_k509 = score_k509.sum(1).unsqueeze(1)
print(score_k509.size())
#vis.plot_scores(img_target, score_k509, figsize=(50,50))
vis.plot_scores(0,score_k509.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 10, 10])
```



In [23]:

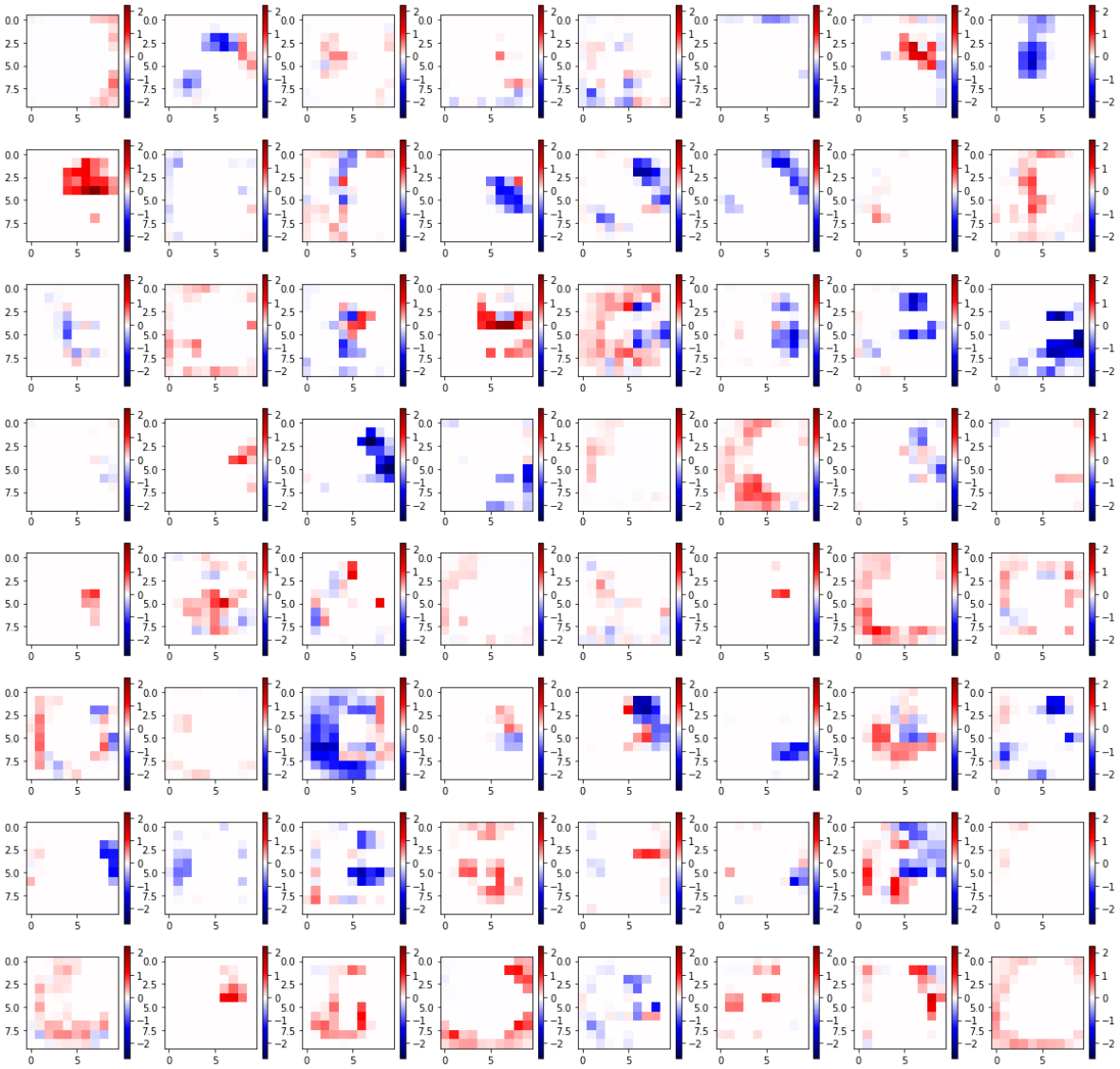
```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k509, 509)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k509
print(score_k509.sum())
print(score_kmapped.sum())
```



```
Max: 0.0004405742511153221, Min: -0.0022416713181883097, Avg: -0.00021104603
988541626, Std:0.0005110064514280787
torch.Size([5, 1, 640, 640])
Variable containing:
-432.2223
[torch.FloatTensor of size 1]

Variable containing:
-432.2223
[torch.FloatTensor of size 1]
```

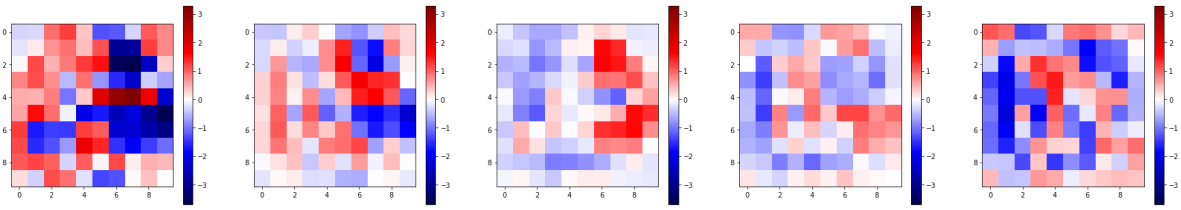
In [24]:

```
vis.plot_scores(img_target, score_rf381)
```



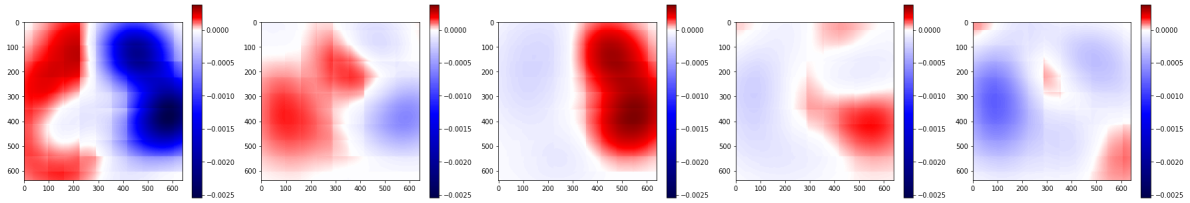
In [25]:

```
#vis.plot_scores(img_target,score_rf381.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf381.sum(1).unsqueeze(0), figsize=(50,50))
```



In [26]:

```
tmp = vis.map_scores_to_input_sz(score_rf381.sum(1).unsqueeze(1), 381)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [27]:

```
layer = 12
rf253_p = layer_vals[layer]
rf253, mp_idx = F.max_pool2d(rf253_p, 2, 2, return_indices = True)  # 1x64x5x5
#del rf253_p
score_rf253, score_k381 = vis.propagate_score_through_conv2d_sz3_pad1(score_rf381, rf253, r
#del score_rf381, val
vis.stats(score_rf253)
```

```
Max: 6.048281192779541, Min: -10.601263999938965, Avg: -0.01247961365675718,
Std:0.3564674965790147
torch.Size([5, 64, 10, 10])
```

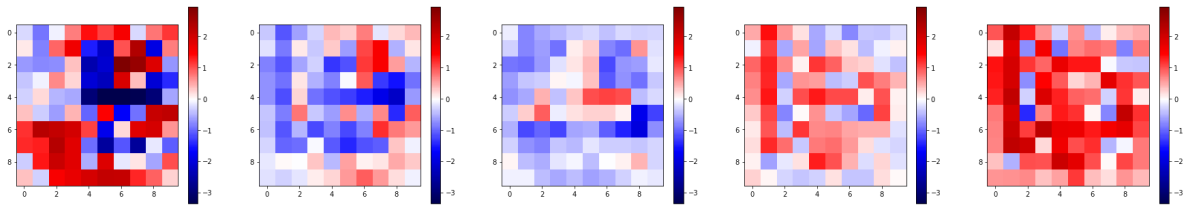
Out[27]:

```
(6.048281192779541,
 -10.601263999938965,
 -0.01247961365675718,
 0.3564674965790147)
```

In [28]:

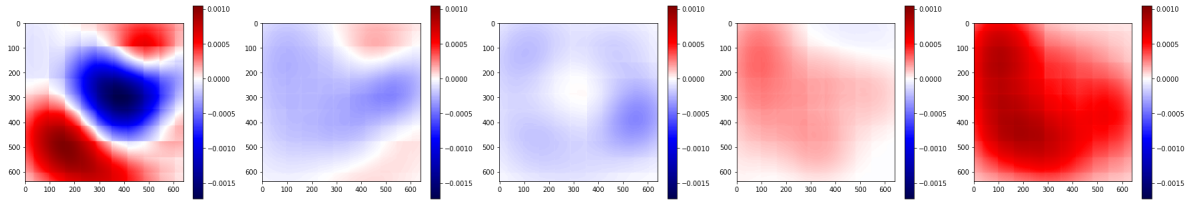
```
score_k381 = score_k381.sum(1).unsqueeze(1)
print(score_k381.size())
#vis.plot_scores(img_target, score_k381, figsize=(50,50))
vis.plot_scores(0,score_k381.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 10, 10])
```



In [29]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k381, 381)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k381
```



```
Max: 0.001045812969096005, Min: -0.0017293045530095696, Avg: 5.1780344940250
715e-05, Std:0.0003852398233642034
torch.Size([5, 1, 640, 640])
```

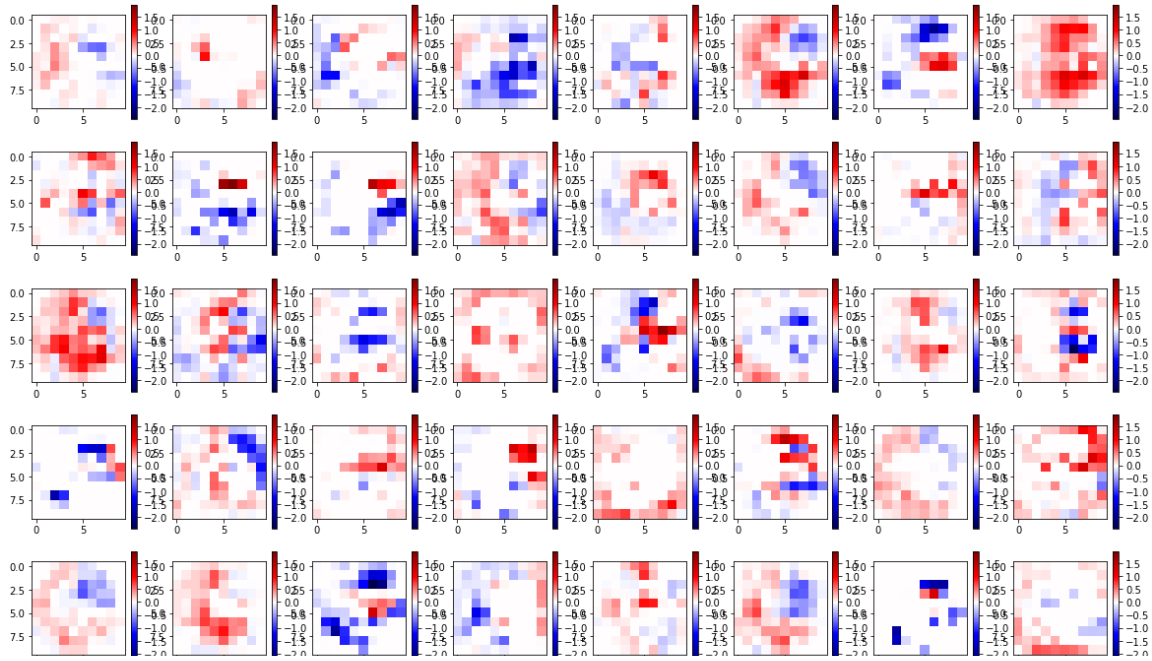
Out[29]:

```
Variable containing:
( 0 , 0 ,.,.) =
 -7.0628e-08 -2.2161e-07 -3.7100e-07  ...   2.6190e-04  2.6050e-04  2.5910
e-04
 -2.3048e-07 -3.8381e-07 -5.3554e-07  ...   2.6334e-04  2.6193e-04  2.6052
e-04
 -3.8986e-07 -5.4555e-07 -6.9959e-07  ...   2.6477e-04  2.6335e-04  2.6193
e-04
                 ...                 ⋱                 ...
  2.5925e-04  2.6123e-04  2.6322e-04  ...   2.6983e-04  2.6839e-04  2.6694
e-04
  2.5802e-04  2.5999e-04  2.6196e-04  ...   2.6942e-04  2.6797e-04  2.6652
e-04
  2.5678e-04  2.5874e-04  2.6070e-04  ...   2.6898e-04  2.6754e-04  2.6609
e-04
      ⋮

( 1 , 0 ,.,.) =
 -5.2021e-05 -5.2635e-05 -5.3247e-05  ...   7.2345e-05  7.1930e-05  7.1517
e-05
 -5.2440e-05 -5.3058e-05 -5.3675e-05  ...   7.2704e-05  7.2287e-05  7.1870
e-05
 -5.2856e-05 -5.3479e-05 -5.4100e-05  ...   7.3062e-05  7.2641e-05  7.2222
e-05
                 ...                 ⋱                 ...
 -4.5717e-06 -4.6775e-06 -4.7825e-06  ...   1.2130e-04  1.2061e-04  1.1991
e-04
 -4.3891e-06 -4.4928e-06 -4.5957e-06  ...   1.2095e-04  1.2025e-04  1.1956
e-04
 -4.2077e-06 -4.3092e-06 -4.4100e-06  ...   1.2058e-04  1.1989e-04  1.1920
e-04
      ⋮

( 2 , 0 ,.,.) =
 -3.7583e-05 -3.8028e-05 -3.8474e-05  ...  -3.7870e-05 -3.7600e-05 -3.7330
e-05
 -3.7935e-05 -3.8384e-05 -3.8833e-05  ...  -3.8136e-05 -3.7864e-05 -3.7592
```

```
e-05
 -3.8286e-05 -3.8739e-05 -3.9192e-05  ...  -3.8401e-05 -3.8128e-05 -3.7854
e-05

                ...                   ⋱                   ...

 -2.2640e-05 -2.2869e-05 -2.3098e-05  ...  -4.3303e-05 -4.3059e-05 -4.2814
e-05
 -2.2496e-05 -2.2723e-05 -2.2950e-05  ...  -4.2939e-05 -4.2698e-05 -4.2455
e-05
 -2.2353e-05 -2.2577e-05 -2.2803e-05  ...  -4.2577e-05 -4.2337e-05 -4.2096
e-05
      ⋮

( 3 , 0 ,.,.) =
  6.1803e-05  6.2578e-05  6.3352e-05  ...  -6.4669e-05 -6.4224e-05 -6.3779
e-05
  6.2253e-05  6.3034e-05  6.3813e-05  ...  -6.5027e-05 -6.4580e-05 -6.4132
e-05
  6.2699e-05  6.3485e-05  6.4269e-05  ...  -6.5384e-05 -6.4934e-05 -6.4484
e-05
                ...                   ⋱                   ...

 -5.4720e-05 -5.5051e-05 -5.5384e-05  ...  -7.6836e-05 -7.6373e-05 -7.5911
e-05
 -5.4582e-05 -5.4913e-05 -5.5246e-05  ...  -7.6549e-05 -7.6088e-05 -7.5627
e-05
 -5.4441e-05 -5.4772e-05 -5.5104e-05  ...  -7.6260e-05 -7.5800e-05 -7.5341
e-05
      ⋮

( 4 , 0 ,.,.) =
  1.4681e-04  1.4875e-04  1.5069e-04  ...  -9.6863e-05 -9.6232e-05 -9.5605
e-05
  1.4790e-04  1.4985e-04  1.5180e-04  ...  -9.7427e-05 -9.6791e-05 -9.6161
e-05
  1.4898e-04  1.5094e-04  1.5290e-04  ...  -9.7994e-05 -9.7355e-05 -9.6720
e-05
                ...                   ⋱                   ...

 -8.1562e-05 -8.1676e-05 -8.1791e-05  ...  -7.3411e-05 -7.3027e-05 -7.2648
e-05
 -8.1730e-05 -8.1850e-05 -8.1970e-05  ...  -7.3574e-05 -7.3189e-05 -7.2810
e-05
 -8.1891e-05 -8.2016e-05 -8.2142e-05  ...  -7.3735e-05 -7.3350e-05 -7.2971
e-05
[torch.FloatTensor of size 5x1x640x640]
```

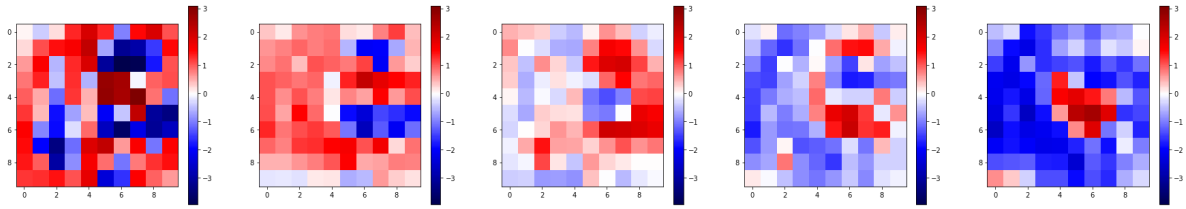
In [30]:

```
vis.plot_scores(img_target,score_rf253)
```



In [31]:

```
#vis.plot_scores(img_target,score_rf253.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf253.sum(1).unsqueeze(0), figsize=(50,50))
```



In [32]:

```
score_mp = vis.propagate_score_through_maxpool_sz2x2_st2x2(score_rf253, mp_idx)
#del score_rf253, mp_idx
layer = 11
rf189 = layer_vals[layer]
score_rf189, score_k253 = vis.propagate_score_through_conv2d_sz3_pad1(score_mp, rf189, rf25
#del score_mp, rf189
vis.stats(score_rf189)
```

```
Max: 3.724321126937866, Min: -10.100293159484863, Avg: -0.002351854353180243
7, Std:0.14117794796852035
torch.Size([5, 64, 20, 20])
```

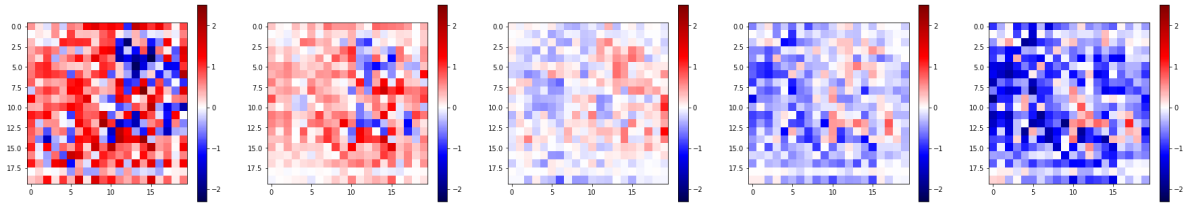
Out[32]:

```
(3.724321126937866,
 -10.100293159484863,
 -0.0023518543531802437,
 0.14117794796852035)
```

In [33]:

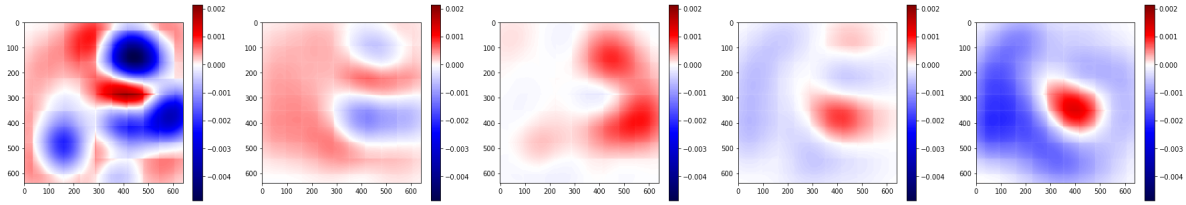
```
score_k253 = score_k253.sum(1).unsqueeze(1)
print(score_k253.size())
#vis.plot_scores(img_target, score_k253, figsize=(50,50))
vis.plot_scores(0,score_k253.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 20, 20])
```



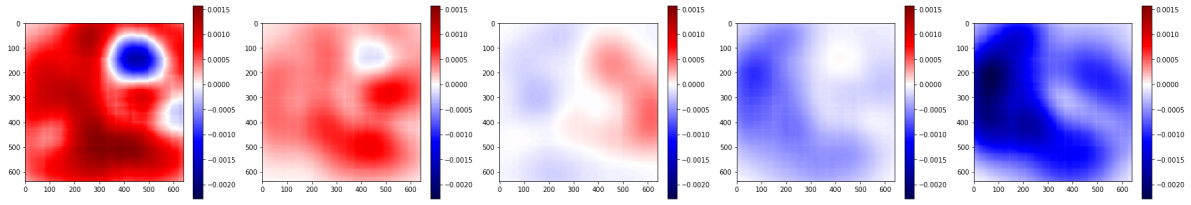
In [34]:

```
#print(score_rf253.view(-1).sum()) # score conservation through mapping to input space test
tmp = vis.map_scores_to_input_sz(score_rf253.sum(1).unsqueeze(1), 253)
#print(tmp.view(-1).sum()) # score conservation through mapping to input space test (2)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [35]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k253, 253)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k253
```



```
Max: 0.001568778301589191, Min: -0.002292977413162589, Avg: -4.8003054505660
92e-05, Std:0.0006753403149404974
torch.Size([5, 1, 640, 640])
```

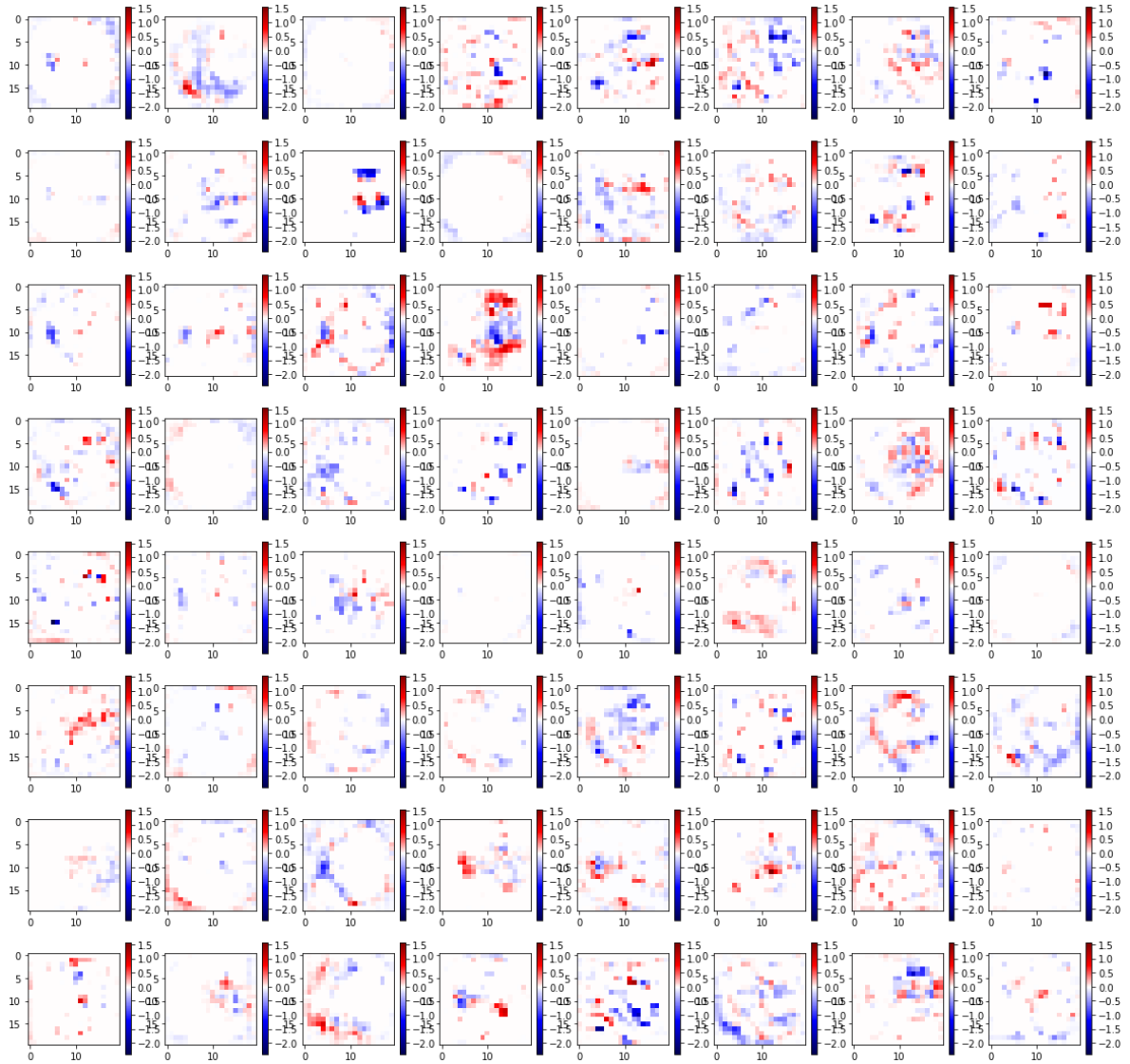
Out[35]:

```
Variable containing:
( 0 , 0 ,.,.) =
  1.3734e-04  1.3836e-04  1.3936e-04  ...   7.1285e-04  7.0750e-04  7.0211
e-04
  1.3856e-04  1.3959e-04  1.4061e-04  ...   7.1891e-04  7.1351e-04  7.0808
e-04
  1.3978e-04  1.4082e-04  1.4185e-04  ...   7.2492e-04  7.1949e-04  7.1401
e-04
                 ...                  ⋱                   ...

  5.2636e-04  5.3145e-04  5.3654e-04  ...   5.8783e-04  5.8389e-04  5.7990
e-04
  5.2224e-04  5.2729e-04  5.3234e-04  ...   5.8364e-04  5.7972e-04  5.7576
e-04
  5.1809e-04  5.2311e-04  5.2811e-04  ...   5.7942e-04  5.7553e-04  5.7159
e-04
      ⋮

( 1 , 0 ,.,.) =
  8.5984e-06  8.4377e-06  8.2700e-06  ...   2.0931e-04  2.0750e-04  2.0569
e-04
  8.7067e-06  8.5455e-06  8.3772e-06  ...   2.1092e-04  2.0910e-04  2.0727
e-04
  8.8121e-06  8.6503e-06  8.4815e-06  ...   2.1252e-04  2.1068e-04  2.0884
e-04
                 ...                  ⋱                   ...

  1.1863e-05  1.1968e-05  1.2074e-05  ...   2.0267e-04  2.0128e-04  1.9988
e-04
  1.1659e-05  1.1763e-05  1.1867e-05  ...   2.0110e-04  1.9972e-04  1.9834
e-04
  1.1459e-05  1.1561e-05  1.1663e-05  ...   1.9953e-04  1.9816e-04  1.9679
e-04
      ⋮

( 2 , 0 ,.,.) =
 -3.8117e-05 -3.8702e-05 -3.9290e-05  ...  -7.2015e-05 -7.1538e-05 -7.1055
e-05
 -3.8572e-05 -3.9164e-05 -3.9759e-05  ...  -7.2663e-05 -7.2182e-05 -7.1695
```

```
e-05
 -3.9029e-05 -3.9628e-05 -4.0230e-05  ...  -7.3307e-05 -7.2823e-05 -7.2332
e-05

                 ...                   ⋱                   ...

 -5.3351e-05 -5.3953e-05 -5.4554e-05  ...  -5.9649e-05 -5.9216e-05 -5.8781
e-05
 -5.2941e-05 -5.3539e-05 -5.4136e-05  ...  -5.9193e-05 -5.8763e-05 -5.8331
e-05
 -5.2528e-05 -5.3121e-05 -5.3714e-05  ...  -5.8734e-05 -5.8307e-05 -5.7878
e-05
      ⋮

( 3 , 0 ,.,.) =
 -8.2766e-05 -8.3157e-05 -8.3531e-05  ...  -1.1120e-04 -1.1044e-04 -1.0967
e-04
 -8.3662e-05 -8.4060e-05 -8.4441e-05  ...  -1.1198e-04 -1.1121e-04 -1.1044
e-04
 -8.4550e-05 -8.4955e-05 -8.5344e-05  ...  -1.1274e-04 -1.1197e-04 -1.1119
e-04
                 ...                   ⋱                   ...

 -8.5746e-05 -8.6510e-05 -8.7275e-05  ...  -1.0373e-04 -1.0299e-04 -1.0224
e-04
 -8.4892e-05 -8.5648e-05 -8.6407e-05  ...  -1.0307e-04 -1.0234e-04 -1.0160
e-04
 -8.4043e-05 -8.4792e-05 -8.5544e-05  ...  -1.0242e-04 -1.0168e-04 -1.0095
e-04
      ⋮

( 4 , 0 ,.,.) =
 -1.6750e-04 -1.6832e-04 -1.6911e-04  ...  -2.1386e-04 -2.1224e-04 -2.1061
e-04
 -1.6927e-04 -1.7010e-04 -1.7090e-04  ...  -2.1526e-04 -2.1362e-04 -2.1198
e-04
 -1.7102e-04 -1.7186e-04 -1.7268e-04  ...  -2.1664e-04 -2.1499e-04 -2.1333
e-04
                 ...                   ⋱                   ...

 -1.2257e-04 -1.2332e-04 -1.2407e-04  ...  -1.7611e-04 -1.7485e-04 -1.7359
e-04
 -1.2128e-04 -1.2202e-04 -1.2277e-04  ...  -1.7520e-04 -1.7395e-04 -1.7270
e-04
 -1.2002e-04 -1.2075e-04 -1.2148e-04  ...  -1.7429e-04 -1.7305e-04 -1.7180
e-04
[torch.FloatTensor of size 5x1x640x640]
```

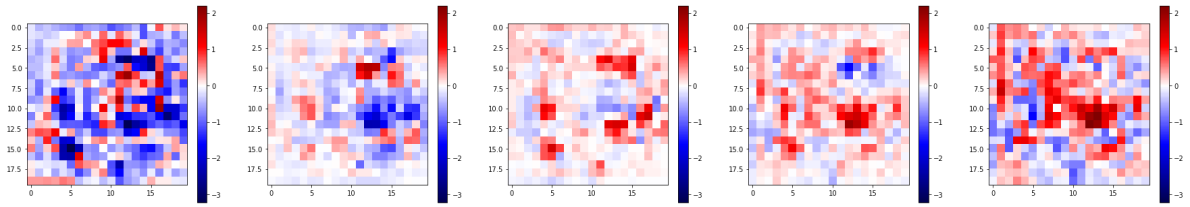
In [36]:

```
vis.plot_scores(img_target,score_rf189)
```



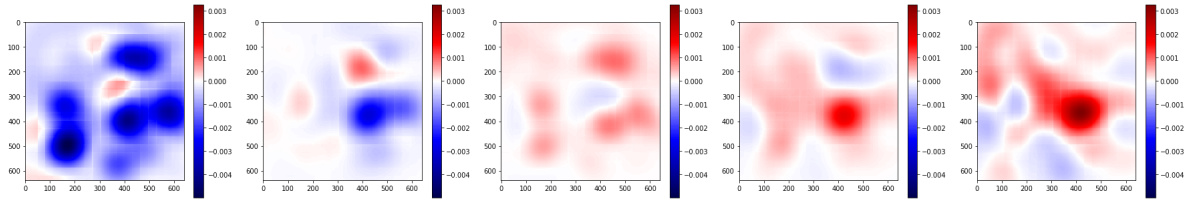
In [37]:

```
#vis.plot_scores(img_target,score_rf189.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf189.sum(1).unsqueeze(0), figsize=(50,50))
```



In [38]:

```
tmp = vis.map_scores_to_input_sz(score_rf189.sum(1).unsqueeze(1), 189)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0, tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [39]:

```
layer = 10
rf125_p = layer_vals[layer]
rf125, mp_idx = F.max_pool2d(rf125_p, 2, 2, return_indices = True)  # 1x64x5x5
#del rf125_p
score_rf125, score_k189 = vis.propagate_score_through_conv2d_sz3_pad1(score_rf189, rf125, r
#del score_rf189,
layer = 10
rf125 = layer_vals[layer]
vis.stats(score_rf125)
```

```
Max: 7.200449466705322, Min: -12.739177703857422, Avg: -0.001125619645060698
7, Std:0.19642142610972496
torch.Size([5, 64, 20, 20])
```

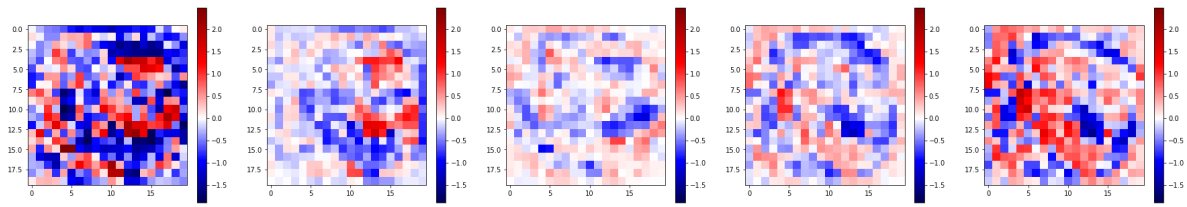
Out[39]:

```
(7.200449466705322,
 -12.739177703857422,
 -0.0011256196450606987,
 0.19642142610972496)
```

In [40]:

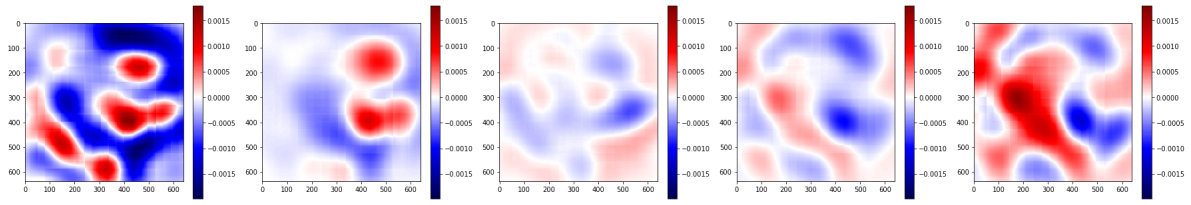
```
score_k189 = score_k189.sum(1).unsqueeze(1)
print(score_k189.size())
#vis.plot_scores(img_target, score_k189, figsize=(50,50))
vis.plot_scores(0,score_k189.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 20, 20])
```



In [41]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k189, 189)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50)
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k189
```



```
Max: 0.001788452616892755, Min: -0.001991763710975647, Avg: -7.6639673695021
06e-05, Std:0.0004457480607013751
torch.Size([5, 1, 640, 640])
```

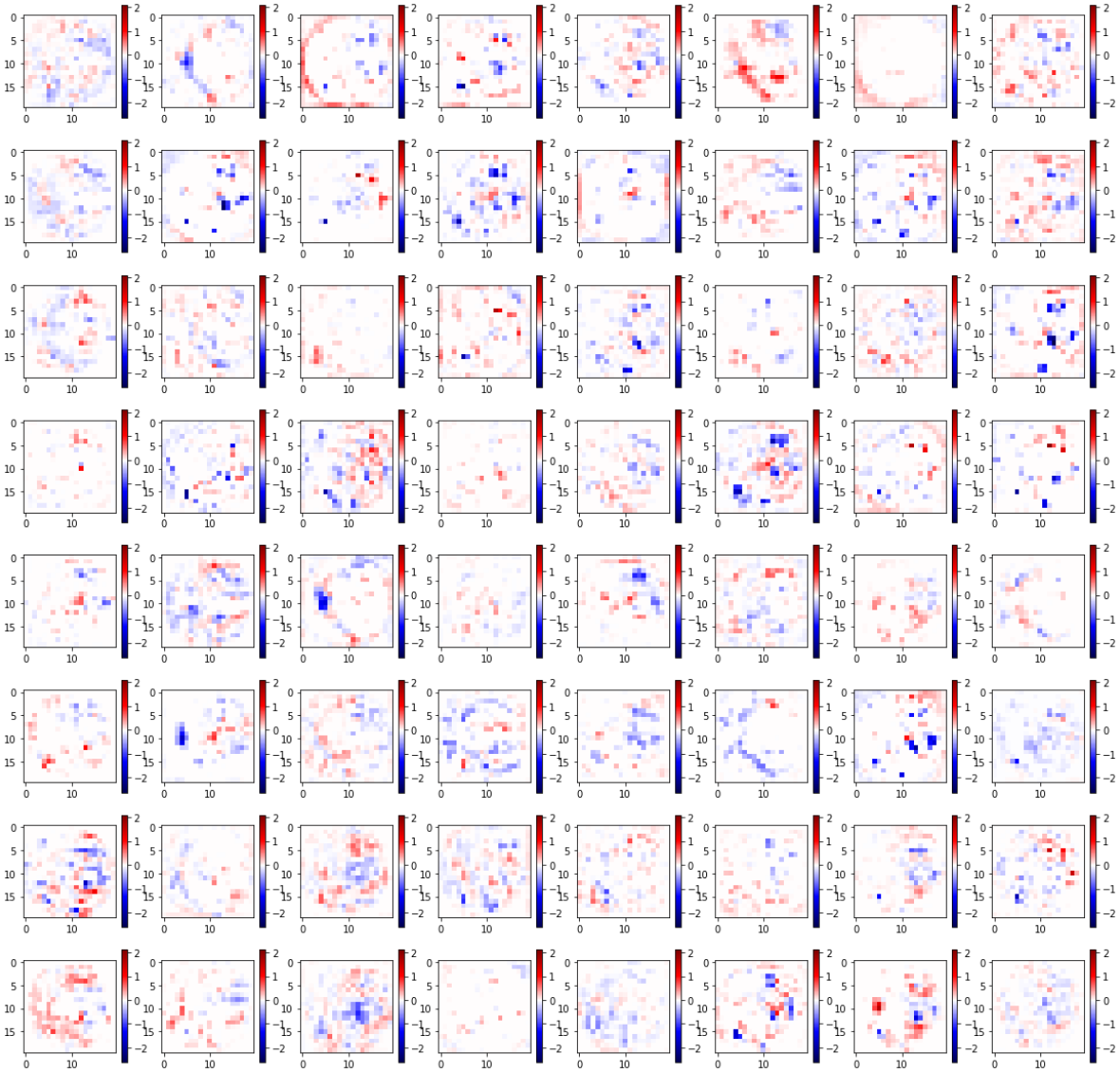
Out[41]:

```
Variable containing:
( 0 , 0 ,.,.) =
  8.1416e-05  8.0716e-05  7.9978e-05  ...   5.1359e-04  5.1181e-04  5.0998
e-04
  8.2018e-05  8.1308e-05  8.0558e-05  ...   5.1447e-04  5.1271e-04  5.1092
e-04
  8.2623e-05  8.1904e-05  8.1144e-05  ...   5.1525e-04  5.1353e-04  5.1177
e-04
                 ...                  ⋱                   ...

  5.1986e-04  5.2562e-04  5.3140e-04  ...   3.5265e-04  3.5155e-04  3.5045
e-04
  5.1705e-04  5.2278e-04  5.2853e-04  ...   3.5137e-04  3.5027e-04  3.4917
e-04
  5.1418e-04  5.1988e-04  5.2560e-04  ...   3.5012e-04  3.4901e-04  3.4791
e-04
      ⋮

( 1 , 0 ,.,.) =
  9.8644e-06  9.1622e-06  8.4350e-06  ...   1.4042e-04  1.3983e-04  1.3922
e-04
  1.0068e-05  9.3592e-06  8.6247e-06  ...   1.4061e-04  1.4002e-04  1.3942
e-04
  1.0273e-05  9.5572e-06  8.8155e-06  ...   1.4077e-04  1.4019e-04  1.3960
e-04
                 ...                  ⋱                   ...

  1.6081e-05  1.6077e-05  1.6068e-05  ...   1.5851e-04  1.5760e-04  1.5670
e-04
  1.5696e-05  1.5692e-05  1.5682e-05  ...   1.5748e-04  1.5658e-04  1.5569
e-04
  1.5319e-05  1.5314e-05  1.5303e-05  ...   1.5647e-04  1.5558e-04  1.5469
e-04
      ⋮

( 2 , 0 ,.,.) =
 -8.7946e-06 -8.9155e-06 -9.0382e-06  ...  -7.6025e-05 -7.5454e-05 -7.4878
e-05
 -8.8580e-06 -8.9791e-06 -9.1022e-06  ...  -7.6442e-05 -7.5869e-05 -7.5291
```

```
e-05
 -8.9259e-06 -9.0474e-06 -9.1707e-06  ...  -7.6846e-05 -7.6271e-05 -7.5691
e-05

                 ...                   ⋱                   ...

 -4.0298e-05 -4.0890e-05 -4.1489e-05  ...  -4.4767e-05 -4.4587e-05 -4.4405
e-05
 -4.0276e-05 -4.0867e-05 -4.1465e-05  ...  -4.4666e-05 -4.4483e-05 -4.4299
e-05
 -4.0244e-05 -4.0834e-05 -4.1431e-05  ...  -4.4559e-05 -4.4374e-05 -4.4187
e-05
      ⋮

( 3 , 0 ,.,.) =
 -5.0737e-05 -4.9512e-05 -4.8234e-05  ...  -9.0401e-05 -8.9689e-05 -8.8983
e-05
 -5.1296e-05 -5.0056e-05 -4.8762e-05  ...  -9.0748e-05 -9.0033e-05 -8.9325
e-05
 -5.1855e-05 -5.0601e-05 -4.9292e-05  ...  -9.1087e-05 -9.0369e-05 -8.9659
e-05
                 ...                   ⋱                   ...

 -1.3001e-04 -1.3116e-04 -1.3231e-04  ...  -8.6554e-05 -8.6053e-05 -8.5554
e-05
 -1.2851e-04 -1.2966e-04 -1.3079e-04  ...  -8.6327e-05 -8.5825e-05 -8.5324
e-05
 -1.2702e-04 -1.2815e-04 -1.2927e-04  ...  -8.6096e-05 -8.5593e-05 -8.5090
e-05
      ⋮

( 4 , 0 ,.,.) =
 -1.4088e-04 -1.3833e-04 -1.3565e-04  ...  -9.7977e-05 -9.7772e-05 -9.7588
e-05
 -1.4230e-04 -1.3971e-04 -1.3699e-04  ...  -9.7860e-05 -9.7662e-05 -9.7485
e-05
 -1.4371e-04 -1.4109e-04 -1.3833e-04  ...  -9.7751e-05 -9.7561e-05 -9.7391
e-05
                 ...                   ⋱                   ...

 -1.9287e-04 -1.9380e-04 -1.9470e-04  ...  -7.6744e-05 -7.6590e-05 -7.6458
e-05
 -1.9022e-04 -1.9114e-04 -1.9203e-04  ...  -7.7810e-05 -7.7641e-05 -7.7494
e-05
 -1.8759e-04 -1.8850e-04 -1.8938e-04  ...  -7.8868e-05 -7.8685e-05 -7.8523
e-05
[torch.FloatTensor of size 5x1x640x640]
```

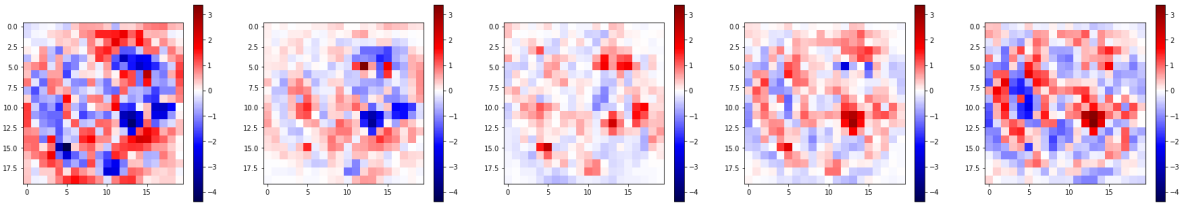
In [42]:

```
vis.plot_scores(img_target,score_rf125)
```



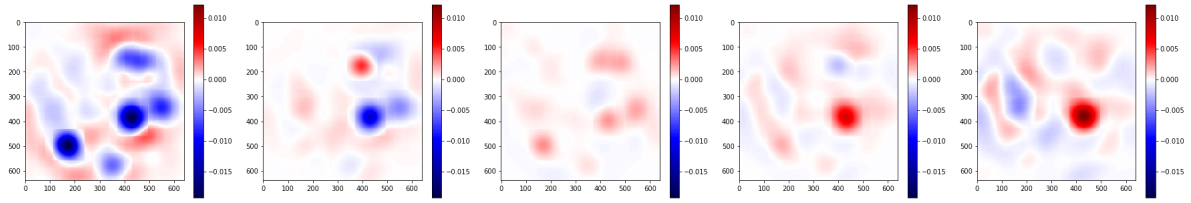
In [43]:

```
#vis.plot_scores(img_target,score_rf125.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf125.sum(1).unsqueeze(0), figsize=(50,50))
```



In [44]:

```
tmp = vis.map_scores_to_input_sz(score_rf125.sum(1).unsqueeze(1), 125)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [45]:

```
score_mp = vis.propagate_score_through_maxpool_sz2x2_st2x2(score_rf125, mp_idx)
#del score_rf125, mp_idx
layer = 9
rf93 = layer_vals[layer]
score_rf93, score_k125 = vis.propagate_score_through_conv2d_sz3_pad1(score_mp, rf93, rf125_
#del score_mp, rf93
vis.stats(score_rf93)
```

```
Max: 5.981862545013428, Min: -10.887755393981934, Avg: -0.00035341310033790
1, Std:0.07917914146139246
torch.Size([5, 64, 40, 40])
```

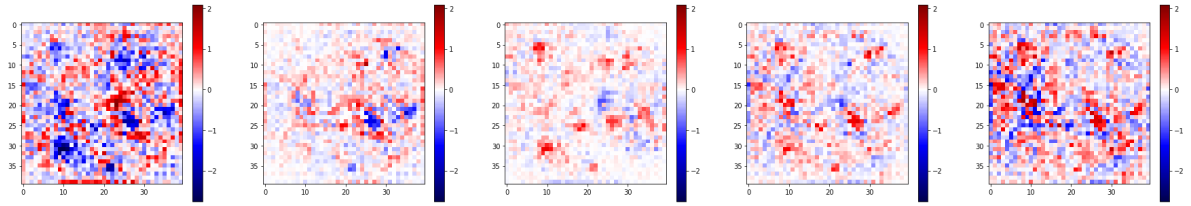
Out[45]:

```
(5.981862545013428,
 -10.887755393981934,
 -0.000353413100337901,
 0.07917914146139246)
```

In [46]:

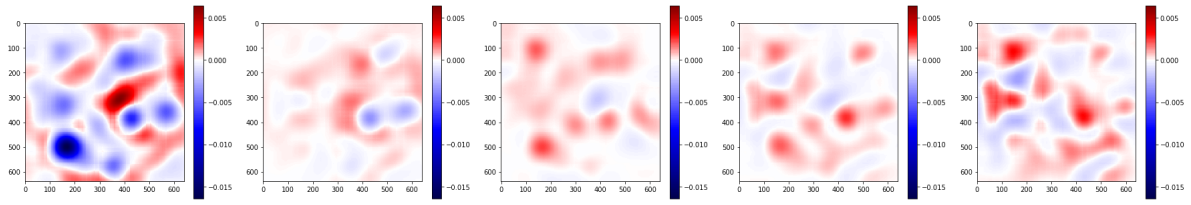
```
score_k125 = score_k125.sum(1).unsqueeze(1)
print(score_k125.size())
#vis.plot_scores(img_target, score_k125, figsize=(50,50))
vis.plot_scores(0,score_k125.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 40, 40])
```



In [47]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k125, 125)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k125
```



```
Max: 0.00643076840788126, Min: -0.016576392576098442, Avg: 1.8002042088119392e-05, Std:0.001188482737313092
torch.Size([5, 1, 640, 640])
```

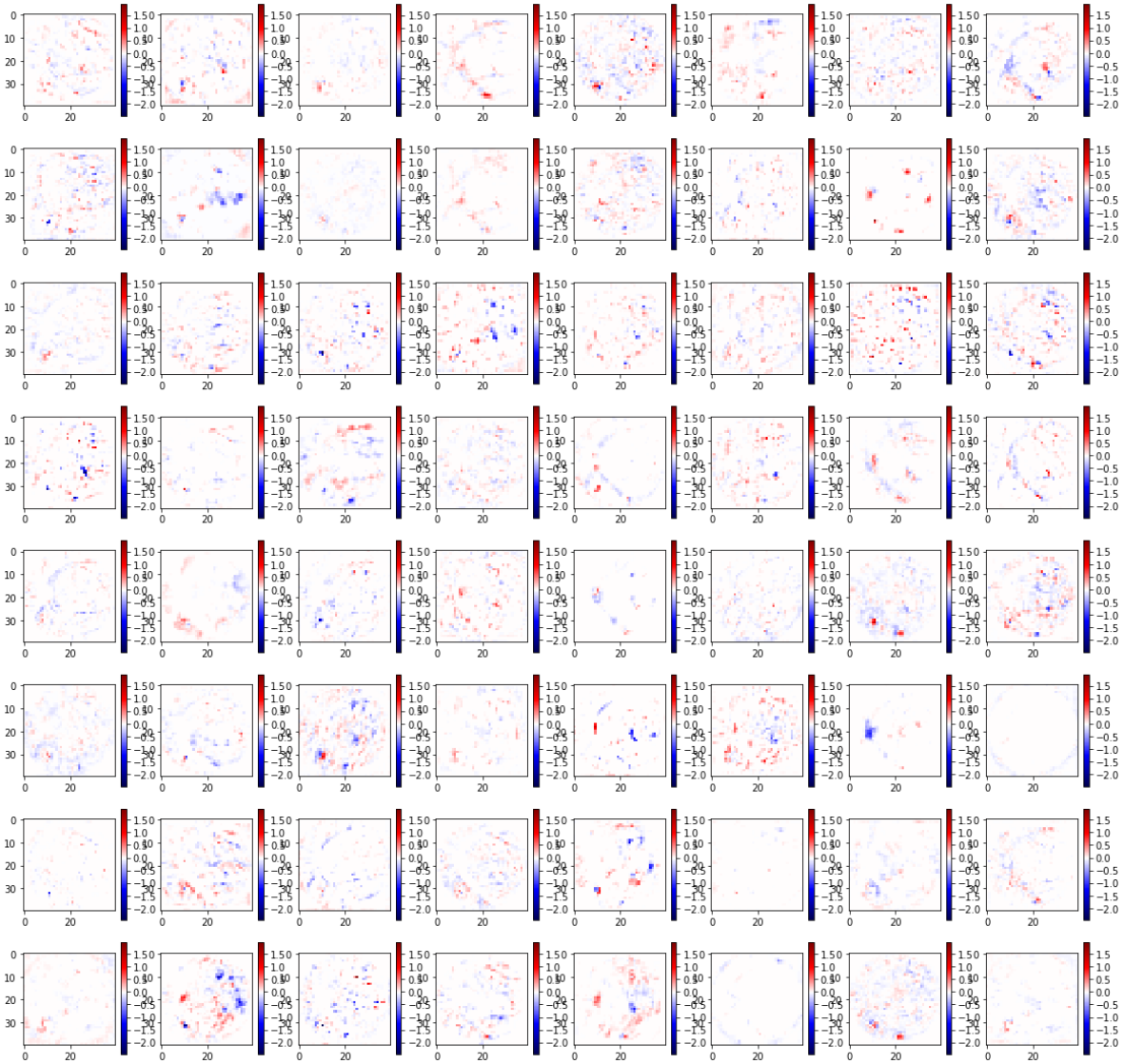
Out[47]:

```
Variable containing:
( 0 , 0 ,.,.) =
 -1.6945e-04 -1.7589e-04 -1.8228e-04  ...   1.1264e-03  1.1160e-03  1.1052e-03
 -1.7287e-04 -1.7943e-04 -1.8593e-04  ...   1.1366e-03  1.1261e-03  1.1152e-03
 -1.7616e-04 -1.8282e-04 -1.8943e-04  ...   1.1464e-03  1.1358e-03  1.1248e-03
                 ...                  ⋱                  ...
  7.0187e-04  7.0645e-04  7.1073e-04  ...   2.1467e-06  6.2487e-06  1.0601e-05
  6.9506e-04  6.9963e-04  7.0391e-04  ...   7.1093e-06  1.1085e-05  1.5309e-05
  6.8816e-04  6.9271e-04  6.9698e-04  ...   1.2263e-05  1.6111e-05  2.0203e-05
      ⋮

( 1 , 0 ,.,.) =
  1.2598e-04  1.2670e-04  1.2731e-04  ...   3.3770e-04  3.3449e-04  3.3116e-04
  1.2806e-04  1.2879e-04  1.2941e-04  ...   3.4041e-04  3.3718e-04  3.3382e-04
  1.3006e-04  1.3081e-04  1.3143e-04  ...   3.4296e-04  3.3970e-04  3.3632e-04
                 ...                  ⋱                  ...
 -1.1254e-05 -1.1815e-05 -1.2372e-05  ...   6.3072e-05  6.3513e-05  6.4021e-05
 -1.1533e-05 -1.2093e-05 -1.2649e-05  ...   6.3480e-05  6.3904e-05  6.4394e-05
 -1.1771e-05 -1.2330e-05 -1.2883e-05  ...   6.3955e-05  6.4361e-05  6.4832e-05
      ⋮

( 2 , 0 ,.,.) =
  3.3515e-04  3.4120e-04  3.4708e-04  ...  -9.7358e-05 -9.6660e-05 -9.5938e-05
  3.4062e-04  3.4679e-04  3.5276e-04  ...  -9.8082e-05 -9.7377e-05 -9.6647
```

```
e-05
  3.4591e-04  3.5217e-04  3.5823e-04  ...  -9.8778e-05 -9.8065e-05 -9.7328
e-05

                ...                   ⋱                   ...

 -8.2812e-05 -8.3512e-05 -8.4164e-05  ...  -3.3333e-05 -3.3523e-05 -3.3712
e-05
 -8.2070e-05 -8.2771e-05 -8.3424e-05  ...  -3.3674e-05 -3.3847e-05 -3.4019
e-05
 -8.1301e-05 -8.2001e-05 -8.2655e-05  ...  -3.4004e-05 -3.4161e-05 -3.4316
e-05
      ⋮

( 3 , 0 ,.,.) =
  6.0586e-05  6.5364e-05  7.0185e-05  ...  -1.5086e-04 -1.4987e-04 -1.4882
e-04
  6.1706e-05  6.6564e-05  7.1464e-05  ...  -1.5163e-04 -1.5062e-04 -1.4957
e-04
  6.2774e-05  6.7709e-05  7.2687e-05  ...  -1.5232e-04 -1.5131e-04 -1.5025
e-04
                ...                   ⋱                   ...

 -1.0802e-04 -1.0832e-04 -1.0859e-04  ...  -2.5338e-05 -2.6133e-05 -2.6952
e-05
 -1.0604e-04 -1.0632e-04 -1.0658e-04  ...  -2.6665e-05 -2.7423e-05 -2.8204
e-05
 -1.0412e-04 -1.0439e-04 -1.0463e-04  ...  -2.7997e-05 -2.8716e-05 -2.9460
e-05
      ⋮

( 4 , 0 ,.,.) =
  1.1341e-04  1.2408e-04  1.3484e-04  ...  -2.8487e-04 -2.8250e-04 -2.8004
e-04
  1.1607e-04  1.2691e-04  1.3787e-04  ...  -2.8660e-04 -2.8422e-04 -2.8174
e-04
  1.1864e-04  1.2966e-04  1.4079e-04  ...  -2.8817e-04 -2.8577e-04 -2.8329
e-04
                ...                   ⋱                   ...

 -9.5353e-05 -9.4185e-05 -9.3013e-05  ...  -4.0367e-05 -4.3131e-05 -4.5848
e-05
 -9.2199e-05 -9.1008e-05 -8.9814e-05  ...  -4.4562e-05 -4.7208e-05 -4.9807
e-05
 -8.9239e-05 -8.8029e-05 -8.6817e-05  ...  -4.8669e-05 -5.1197e-05 -5.3678
e-05
[torch.FloatTensor of size 5x1x640x640]
```

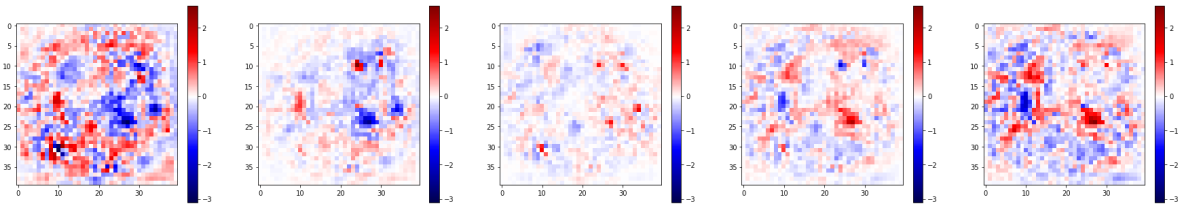
In [48]:

```
vis.plot_scores(img_target,score_rf93)
```



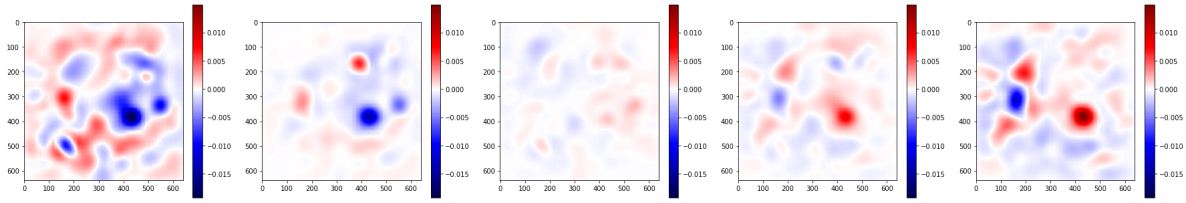
In [49]:

```
#vis.plot_scores(img_target,score_rf93.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf93.sum(1).unsqueeze(0), figsize=(50,50))
```



In [50]:

```
tmp = vis.map_scores_to_input_sz(score_rf93.sum(1).unsqueeze(1), 93)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [51]:

```
layer = 8
rf61_p = layer_vals[layer]
rf61, mp_idx = F.max_pool2d(rf61_p, 2, 2, return_indices = True)  # 1x64x5x5
score_rf61, score_k93 = vis.propagate_score_through_conv2d_sz3_pad1(score_rf93, rf61, rf93,
#del score_rf93, val
vis.stats(score_rf61)
```

```
Max: 7.401787281036377, Min: -18.24847412109375, Avg: -0.0003375353735313160
4, Std:0.10855066936173227
torch.Size([5, 64, 40, 40])
```

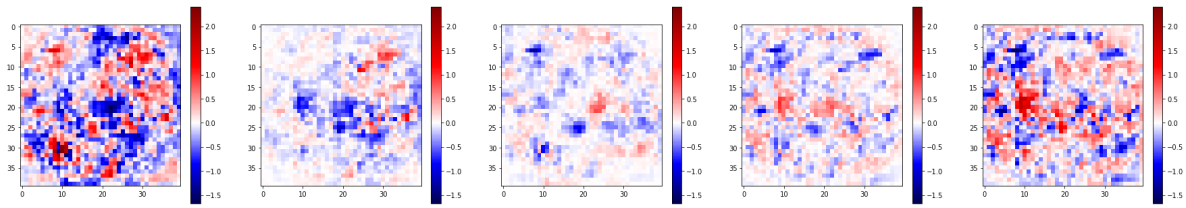
Out[51]:

```
(7.401787281036377,
 -18.24847412109375,
 -0.00033753537353131604,
 0.10855066936173227)
```

In [52]:

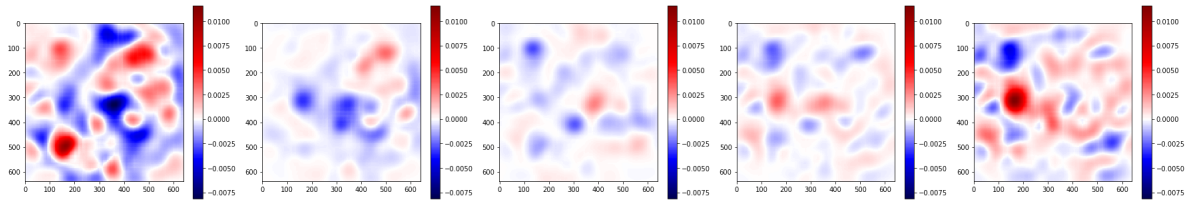
```
score_k93 = score_k93.sum(1).unsqueeze(1)
print(score_k93.size())
#vis.plot_scores(img_target, score_k93, figsize=(50,50))
vis.plot_scores(0,score_k93.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 40, 40])
```



In [53]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k93, 93)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k93
```



```
Max: 0.011636764742434025, Min: -0.008191878907382488, Avg: -3.9694357402996
83e-06, Std:0.0012354762757171257
torch.Size([5, 1, 640, 640])
```

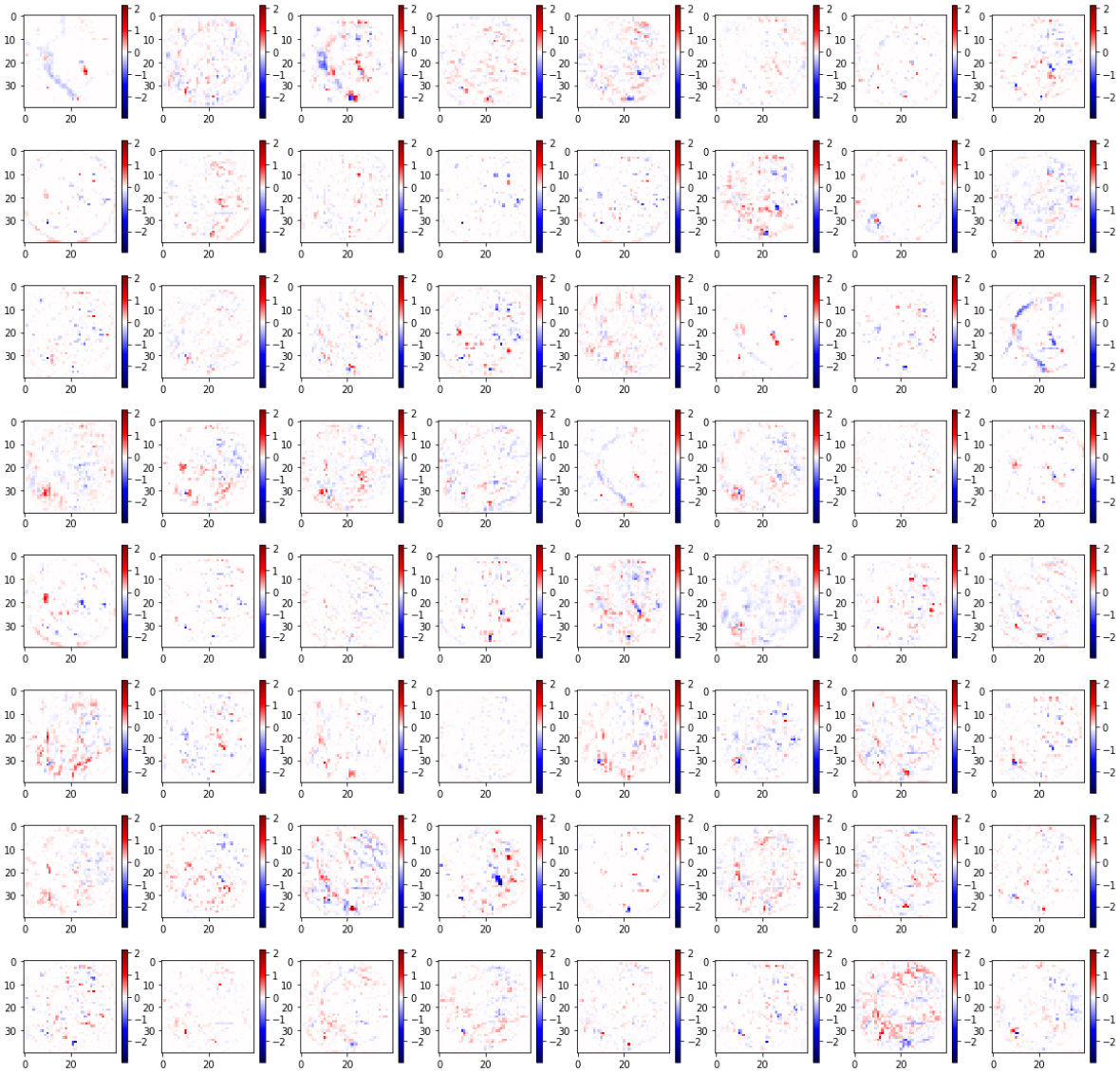
Out[53]:

```
Variable containing:
( 0 , 0 ,.,.) =
  3.1475e-05  3.3072e-05  3.4713e-05  ...   5.7373e-04  5.7872e-04  5.8372
e-04
  3.4023e-05  3.5762e-05  3.7546e-05  ...   5.6708e-04  5.7245e-04  5.7783
e-04
  3.6565e-05  3.8445e-05  4.0370e-05  ...   5.6026e-04  5.6599e-04  5.7175
e-04
                 ...                  ⋱                  ...

  6.1500e-04  6.1925e-04  6.2347e-04  ...   7.8815e-05  8.1052e-05  8.3459
e-05
  6.1168e-04  6.1595e-04  6.2021e-04  ...   7.8323e-05  8.0619e-05  8.3083
e-05
  6.0824e-04  6.1255e-04  6.1683e-04  ...   7.8222e-05  8.0567e-05  8.3077
e-05
      ⋮

( 1 , 0 ,.,.) =
  9.5382e-05  9.4909e-05  9.4328e-05  ...   1.5285e-04  1.5501e-04  1.5716
e-04
  9.6750e-05  9.6266e-05  9.5673e-05  ...   1.5061e-04  1.5288e-04  1.5515
e-04
  9.8089e-05  9.7594e-05  9.6988e-05  ...   1.4835e-04  1.5073e-04  1.5311
e-04
                 ...                  ⋱                  ...

  2.3818e-05  2.4185e-05  2.4523e-05  ...   7.9013e-05  7.9105e-05  7.9241
e-05
  2.2858e-05  2.3210e-05  2.3534e-05  ...   7.8468e-05  7.8575e-05  7.8726
e-05
  2.1893e-05  2.2229e-05  2.2539e-05  ...   7.8016e-05  7.8137e-05  7.8301
e-05
      ⋮

( 2 , 0 ,.,.) =
  1.3114e-04  1.2936e-04  1.2742e-04  ...  -7.4441e-05 -7.4393e-05 -7.4342
e-05
  1.3103e-04  1.2913e-04  1.2707e-04  ...  -7.4337e-05 -7.4311e-05 -7.4282
```

```
e-05
  1.3091e-04  1.2889e-04  1.2670e-04  ...  -7.4207e-05 -7.4203e-05 -7.4195
e-05

                ...                   ⋱                   ...

 -7.5548e-05 -7.6366e-05 -7.7174e-05  ...  -5.0368e-05 -4.9909e-05 -4.9455
e-05
 -7.5151e-05 -7.5971e-05 -7.6781e-05  ...  -5.0026e-05 -4.9578e-05 -4.9134
e-05
 -7.4721e-05 -7.5542e-05 -7.6354e-05  ...  -4.9681e-05 -4.9243e-05 -4.8808
e-05
      ⋮

( 3 , 0 ,.,.) =
 -9.1265e-05 -9.2402e-05 -9.3478e-05  ...  -1.3358e-04 -1.3280e-04 -1.3201
e-04
 -9.4609e-05 -9.5858e-05 -9.7047e-05  ...  -1.3444e-04 -1.3364e-04 -1.3283
e-04
 -9.7903e-05 -9.9263e-05 -1.0056e-04  ...  -1.3529e-04 -1.3446e-04 -1.3363
e-04
                ...                   ⋱                   ...

 -1.9404e-04 -1.9653e-04 -1.9892e-04  ...  -7.1124e-05 -7.0262e-05 -6.9433
e-05
 -1.9063e-04 -1.9309e-04 -1.9545e-04  ...  -7.0557e-05 -6.9727e-05 -6.8928
e-05
 -1.8716e-04 -1.8959e-04 -1.9191e-04  ...  -7.0013e-05 -6.9213e-05 -6.8444
e-05
      ⋮

( 4 , 0 ,.,.) =
 -2.5086e-04 -2.5371e-04 -2.5637e-04  ...  -1.7933e-04 -1.7945e-04 -1.7959
e-04
 -2.5832e-04 -2.6141e-04 -2.6430e-04  ...  -1.7976e-04 -1.7987e-04 -1.7999
e-04
 -2.6558e-04 -2.6891e-04 -2.7204e-04  ...  -1.8024e-04 -1.8033e-04 -1.8043
e-04
                ...                   ⋱                   ...

 -2.9340e-04 -2.9688e-04 -3.0015e-04  ...  -7.3915e-05 -7.3915e-05 -7.4008
e-05
 -2.8672e-04 -2.9013e-04 -2.9333e-04  ...  -7.3999e-05 -7.4086e-05 -7.4262
e-05
 -2.7995e-04 -2.8329e-04 -2.8642e-04  ...  -7.4181e-05 -7.4349e-05 -7.4602
e-05
[torch.FloatTensor of size 5x1x640x640]
```

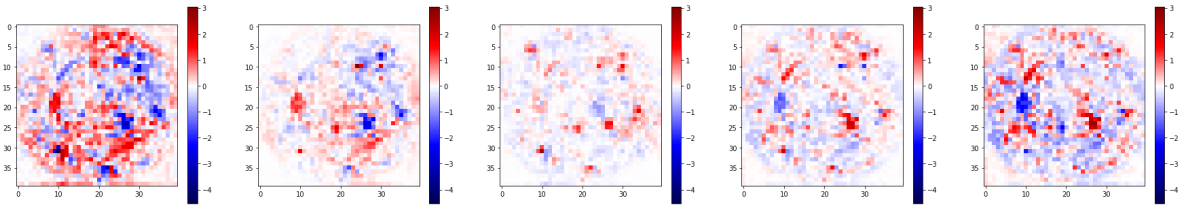
In [54]:

```
vis.plot_scores(img_target,score_rf61)
```



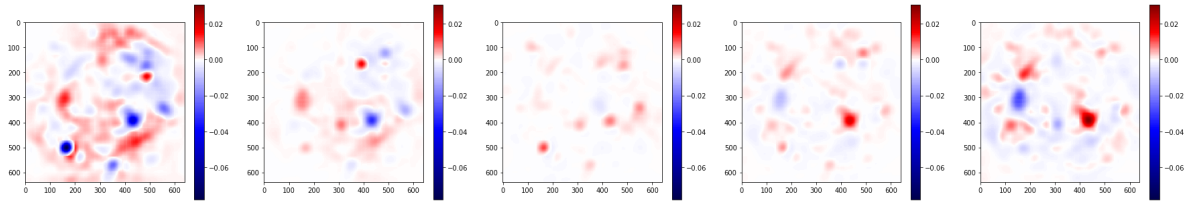
In [55]:

```
#vis.plot_scores(img_target,score_rf61.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf61.sum(1).unsqueeze(0), figsize=(50,50))
```



In [56]:

```
tmp = vis.map_scores_to_input_sz(score_rf61.sum(1).unsqueeze(1), 61)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [57]:

```
score_mp = vis.propagate_score_through_maxpool_sz2x2_st2x2(score_rf61, mp_idx)
#del score_rf61, mp_idx
layer = 7
rf45 = layer_vals[layer]
score_rf45, score_k61 = vis.propagate_score_through_conv2d_sz3_pad1(score_mp, rf45, rf61_p,
#del score_mp, rf45
vis.stats(score_rf45)
```

```
Max: 12.553932189941406, Min: -17.085023880004883, Avg: 5.2998503565246254e-
05, Std:0.051321961675574096
torch.Size([5, 64, 80, 80])
```

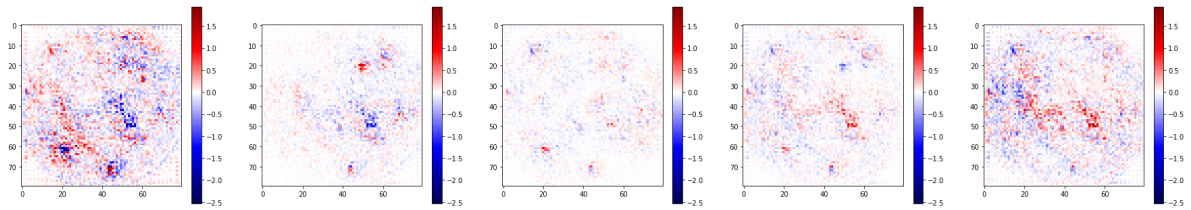
Out[57]:

```
(12.553932189941406,
 -17.085023880004883,
 5.2998503565246254e-05,
 0.051321961675574096)
```

In [58]:

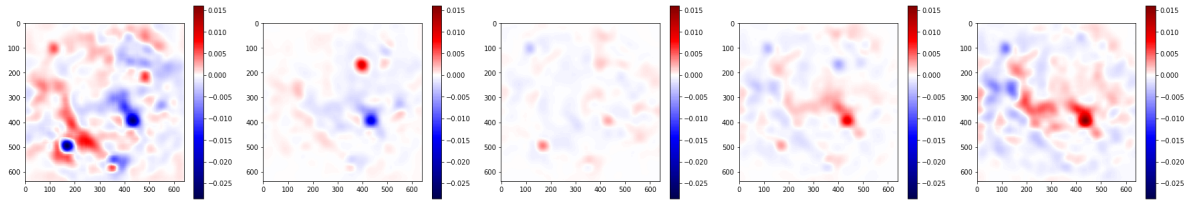
```
score_k61 = score_k61.sum(1).unsqueeze(1)
print(score_k61.size())
#vis.plot_scores(img_target, score_k61, figsize=(50,50))
vis.plot_scores(0,score_k61.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 80, 80])
```



In [59]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k61, 61)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k61
```



```
Max: 0.01612282544374466, Min: -0.02901856228709221, Avg: -0.000137382347334
44944, Std:0.0015912877834708438
torch.Size([5, 1, 640, 640])
```

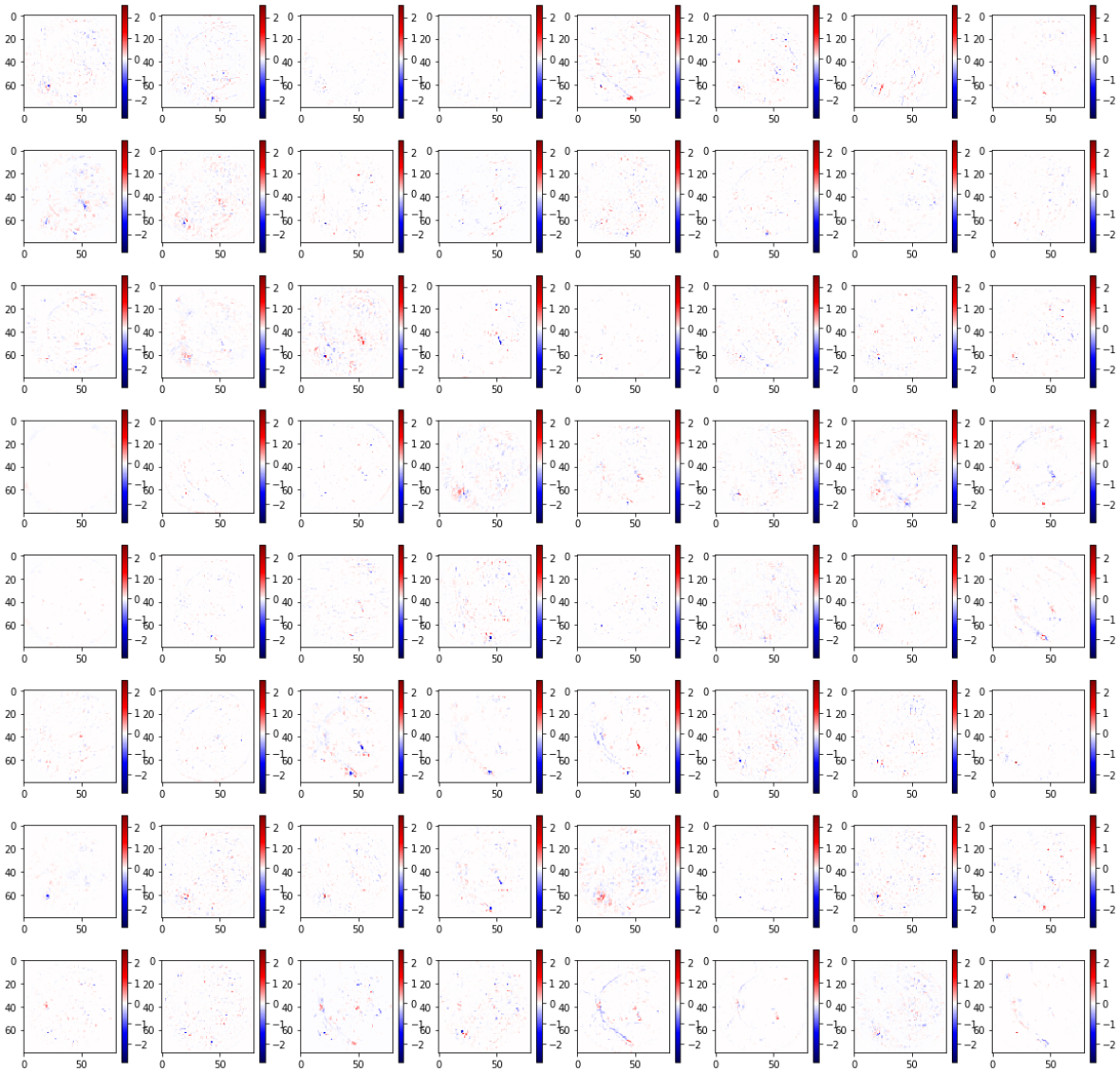
Out[59]:

```
Variable containing:
( 0 , 0 ,.,.) =
  2.3094e-05  2.4432e-05  2.5825e-05  ...   7.8045e-04  7.8199e-04  7.8284
e-04
  2.5890e-05  2.7374e-05  2.8911e-05  ...   7.8090e-04  7.8293e-04  7.8422
e-04
  2.8750e-05  3.0379e-05  3.2061e-05  ...   7.8071e-04  7.8326e-04  7.8502
e-04
                 ...                  ⋱                  ...

  7.3407e-04  7.4520e-04  7.5633e-04  ...   4.9375e-04  4.7497e-04  4.5613
e-04
  7.2722e-04  7.3817e-04  7.4911e-04  ...   4.7393e-04  4.5621e-04  4.3844
e-04
  7.1992e-04  7.3066e-04  7.4138e-04  ...   4.5404e-04  4.3739e-04  4.2069
e-04
      ⋮

( 1 , 0 ,.,.) =
  1.0018e-04  9.9808e-05  9.9331e-05  ...   2.4245e-04  2.4243e-04  2.4217
e-04
  1.0100e-04  1.0057e-04  1.0003e-04  ...   2.4398e-04  2.4406e-04  2.4386
e-04
  1.0172e-04  1.0121e-04  1.0061e-04  ...   2.4535e-04  2.4551e-04  2.4540
e-04
                 ...                  ⋱                  ...

 -2.1538e-05 -2.3202e-05 -2.4800e-05  ...   1.7855e-04  1.7329e-04  1.6804
e-04
 -2.0652e-05 -2.2244e-05 -2.3772e-05  ...   1.7354e-04  1.6854e-04  1.6356
e-04
 -1.9709e-05 -2.1226e-05 -2.2681e-05  ...   1.6849e-04  1.6376e-04  1.5904
e-04
      ⋮

( 2 , 0 ,.,.) =
  1.2330e-04  1.2057e-04  1.1770e-04  ...  -8.7617e-05 -8.7017e-05 -8.6398
e-05
  1.2090e-04  1.1790e-04  1.1475e-04  ...  -8.8045e-05 -8.7448e-05 -8.6832
```

```
e-05
  1.1834e-04  1.1506e-04  1.1164e-04  ...  -8.8418e-05 -8.7826e-05 -8.7215
e-05

                ...                   ⋱                   ...

 -8.3624e-05 -8.5144e-05 -8.6687e-05  ...  -6.3592e-05 -6.2700e-05 -6.1784
e-05
 -8.2933e-05 -8.4427e-05 -8.5943e-05  ...  -6.2471e-05 -6.1622e-05 -6.0750
e-05
 -8.2186e-05 -8.3652e-05 -8.5138e-05  ...  -6.1349e-05 -6.0543e-05 -5.9714
e-05
      ⋮

( 3 , 0 ,.,.) =
 -8.9796e-05 -9.1356e-05 -9.2905e-05  ...  -1.7552e-04 -1.7287e-04 -1.7018
e-04
 -9.3300e-05 -9.4981e-05 -9.6651e-05  ...  -1.7873e-04 -1.7598e-04 -1.7318
e-04
 -9.6770e-05 -9.8569e-05 -1.0036e-04  ...  -1.8188e-04 -1.7903e-04 -1.7613
e-04
                ...                   ⋱                   ...

 -7.6973e-05 -7.5211e-05 -7.3685e-05  ...  -9.4179e-05 -9.1929e-05 -8.9721
e-05
 -7.7810e-05 -7.6161e-05 -7.4739e-05  ...  -9.2067e-05 -8.9953e-05 -8.7878
e-05
 -7.8776e-05 -7.7246e-05 -7.5934e-05  ...  -8.9998e-05 -8.8017e-05 -8.6072
e-05
      ⋮

( 4 , 0 ,.,.) =
 -2.5276e-04 -2.5735e-04 -2.6188e-04  ...  -2.8340e-04 -2.8043e-04 -2.7724
e-04
 -2.6140e-04 -2.6631e-04 -2.7116e-04  ...  -2.8900e-04 -2.8592e-04 -2.8261
e-04
 -2.6991e-04 -2.7513e-04 -2.8030e-04  ...  -2.9453e-04 -2.9135e-04 -2.8792
e-04
                ...                   ⋱                   ...

 -4.1292e-05 -3.5049e-05 -2.9257e-05  ...  -2.7799e-04 -2.6886e-04 -2.5960
e-04
 -4.3901e-05 -3.7941e-05 -3.2414e-05  ...  -2.7056e-04 -2.6184e-04 -2.5301
e-04
 -4.6850e-05 -4.1185e-05 -3.5936e-05  ...  -2.6284e-04 -2.5456e-04 -2.4615
e-04
[torch.FloatTensor of size 5x1x640x640]
```

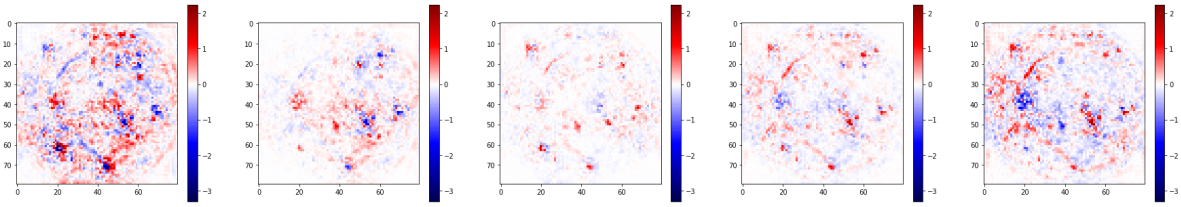
In [60]:

```
vis.plot_scores(img_target,score_rf45)
```



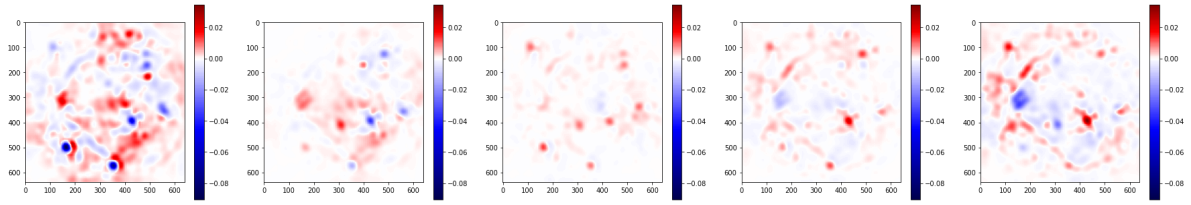
In [61]:

```
#vis.plot_scores(img_target,score_rf45.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf45.sum(1).unsqueeze(0), figsize=(50,50))
```



In [62]:

```
tmp = vis.map_scores_to_input_sz(score_rf45.sum(1).unsqueeze(1), 45)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [63]:

```
layer = 6
rf29_p = layer_vals[layer]
rf29, mp_idx = F.max_pool2d(rf29_p, 2, 2, return_indices = True)  # 1x64x5x5
#del rf29_p
score_rf29, score_k45 = vis.propagate_score_through_conv2d_sz3_pad1(score_rf45, rf29, rf45,
#del score_rf45, val
vis.stats(score_rf29)
```

```
Max: 16.318115234375, Min: -21.605670928955078, Avg: -2.3712126556145256e-0
5, Std:0.0685936251277833
torch.Size([5, 64, 80, 80])
```

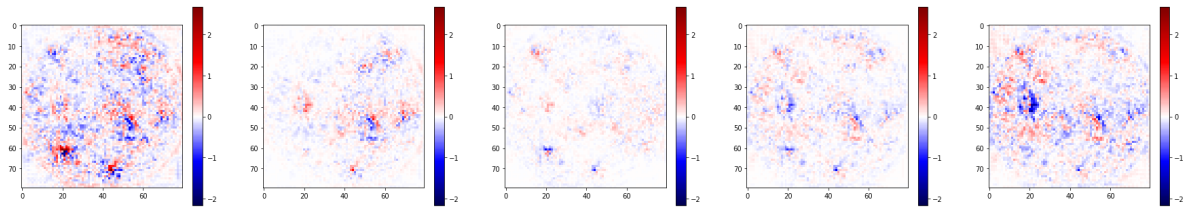
Out[63]:

```
(16.318115234375,
 -21.605670928955078,
 -2.3712126556145256e-05,
 0.0685936251277833)
```

In [64]:

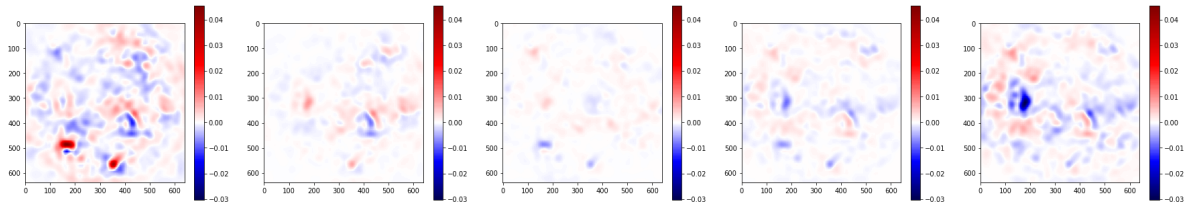
```
score_k45 = score_k45.sum(1).unsqueeze(1)
print(score_k45.size())
#vis.plot_scores(img_target, score_k45, figsize=(50,50))
vis.plot_scores(0,score_k45.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 80, 80])
```



In [65]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k45.sum(1).unsqueeze(1), 45)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k45
```



```
Max: 0.04638511687517166, Min: -0.030861297622323036, Avg: 7.671062200151175
e-05, Std:0.0019549206248127005
torch.Size([5, 1, 640, 640])
```

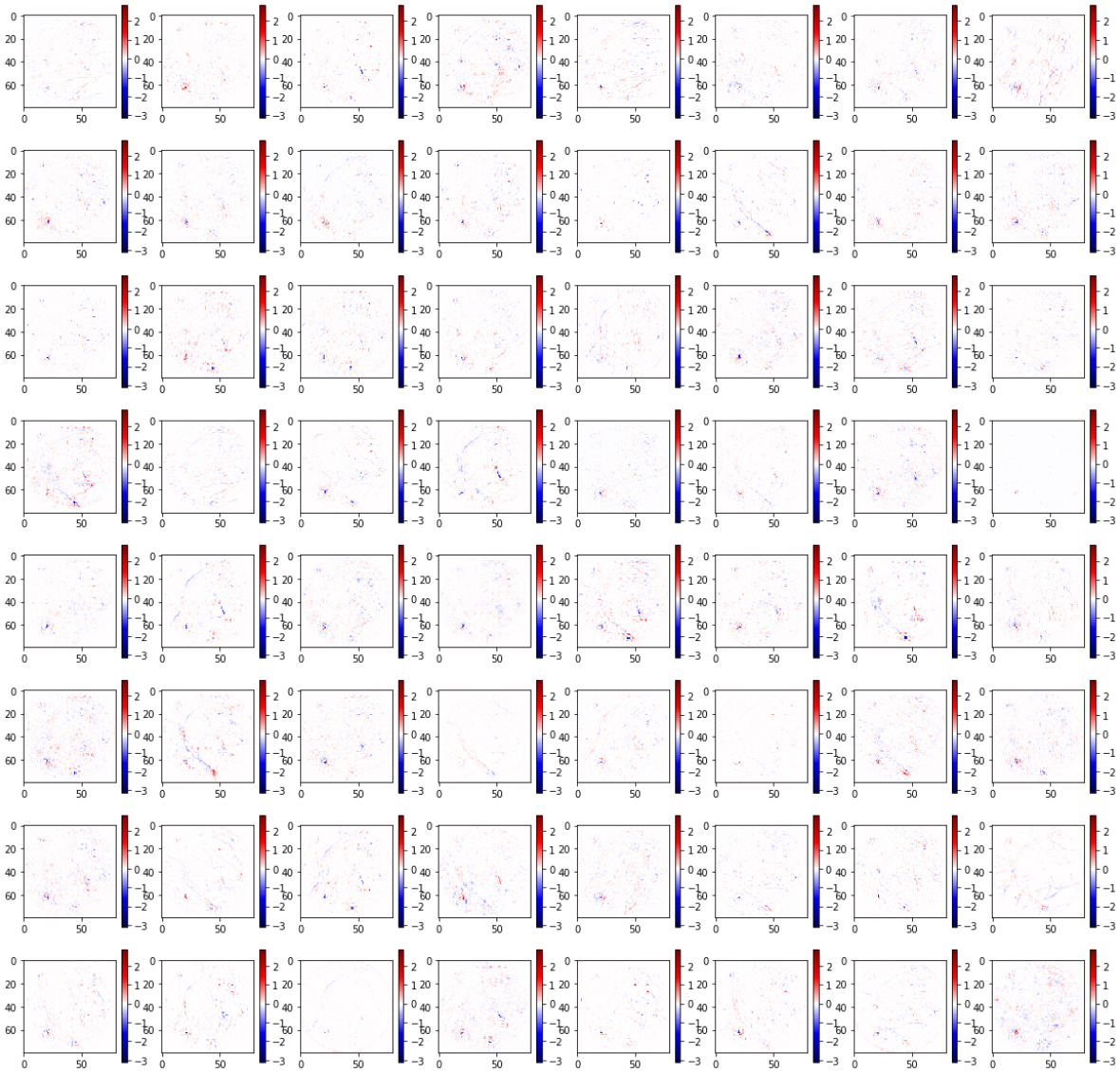
Out[65]:

```
Variable containing:
( 0 , 0 ,.,.) =
 -4.2532e-05 -4.4479e-05 -4.5952e-05  ...   7.9601e-04  7.9146e-04  7.8696
e-04
 -4.4746e-05 -4.6677e-05 -4.8079e-05  ...   8.0985e-04  8.0437e-04  7.9896
e-04
 -4.6750e-05 -4.8638e-05 -4.9939e-05  ...   8.2483e-04  8.1830e-04  8.1187
e-04
                 ...                 ⋱                 ...

  7.4600e-04  7.5732e-04  7.6866e-04  ...   3.1560e-04  3.0985e-04  3.0381
e-04
  7.3688e-04  7.4786e-04  7.5886e-04  ...   3.0886e-04  3.0327e-04  2.9740
e-04
  7.2752e-04  7.3816e-04  7.4881e-04  ...   3.0186e-04  2.9645e-04  2.9076
e-04
      ⋮

( 1 , 0 ,.,.) =
  9.2875e-05  9.1608e-05  9.0266e-05  ...   2.5794e-04  2.5510e-04  2.5226
e-04
  9.3606e-05  9.2268e-05  9.0858e-05  ...   2.6430e-04  2.6101e-04  2.5774
e-04
  9.4322e-05  9.2920e-05  9.1448e-05  ...   2.7101e-04  2.6724e-04  2.6350
e-04
                 ...                 ⋱                 ...

 -3.9552e-05 -4.1989e-05 -4.4216e-05  ...   1.2574e-04  1.2431e-04  1.2283
e-04
 -3.8325e-05 -4.0709e-05 -4.2896e-05  ...   1.2446e-04  1.2303e-04  1.2156
e-04
 -3.6893e-05 -3.9211e-05 -4.1343e-05  ...   1.2312e-04  1.2170e-04  1.2024
e-04
      ⋮

( 2 , 0 ,.,.) =
  1.8628e-04  1.8713e-04  1.8758e-04  ...  -9.0263e-05 -8.9491e-05 -8.8701
e-05
  1.8881e-04  1.8961e-04  1.8997e-04  ...  -9.1568e-05 -9.0745e-05 -8.9902
```

```
e-05
  1.9109e-04  1.9181e-04  1.9206e-04  ...  -9.2911e-05 -9.2034e-05 -9.1134
e-05

                ...                   ⋱                   ...

 -1.0173e-04 -1.0410e-04 -1.0641e-04  ...  -6.6957e-05 -6.5798e-05 -6.4622
e-05
 -1.0007e-04 -1.0236e-04 -1.0460e-04  ...  -6.5806e-05 -6.4697e-05 -6.3571
e-05
 -9.8296e-05 -1.0052e-04 -1.0268e-04  ...  -6.4611e-05 -6.3554e-05 -6.2480
e-05
      ⋮

( 3 , 0 ,.,.) =
 -1.9343e-05 -1.6542e-05 -1.4106e-05  ...  -1.6605e-04 -1.6274e-04 -1.5960
e-04
 -1.7997e-05 -1.5123e-05 -1.2662e-05  ...  -1.6954e-04 -1.6601e-04 -1.6266
e-04
 -1.6855e-05 -1.3937e-05 -1.1480e-05  ...  -1.7313e-04 -1.6937e-04 -1.6580
e-04
                ...                   ⋱                   ...

 -1.1136e-04 -1.1084e-04 -1.1037e-04  ...  -1.1308e-04 -1.0987e-04 -1.0662
e-04
 -1.1005e-04 -1.0955e-04 -1.0908e-04  ...  -1.1047e-04 -1.0742e-04 -1.0433
e-04
 -1.0884e-04 -1.0835e-04 -1.0790e-04  ...  -1.0774e-04 -1.0486e-04 -1.0194
e-04
      ⋮

( 4 , 0 ,.,.) =
 -8.1771e-05 -7.4407e-05 -6.7716e-05  ...  -2.6739e-04 -2.6137e-04 -2.5577
e-04
 -7.9265e-05 -7.1683e-05 -6.4868e-05  ...  -2.7444e-04 -2.6784e-04 -2.6169
e-04
 -7.7317e-05 -6.9591e-05 -6.2732e-05  ...  -2.8184e-04 -2.7462e-04 -2.6789
e-04
                ...                   ⋱                   ...

 -8.2742e-05 -7.8150e-05 -7.3849e-05  ...  -1.0221e-04 -9.8859e-05 -9.6343
e-05
 -8.1937e-05 -7.7409e-05 -7.3158e-05  ...  -1.0243e-04 -9.9390e-05 -9.7120
e-05
 -8.1559e-05 -7.7123e-05 -7.2949e-05  ...  -1.0305e-04 -1.0029e-04 -9.8232
e-05
[torch.FloatTensor of size 5x1x640x640]
```

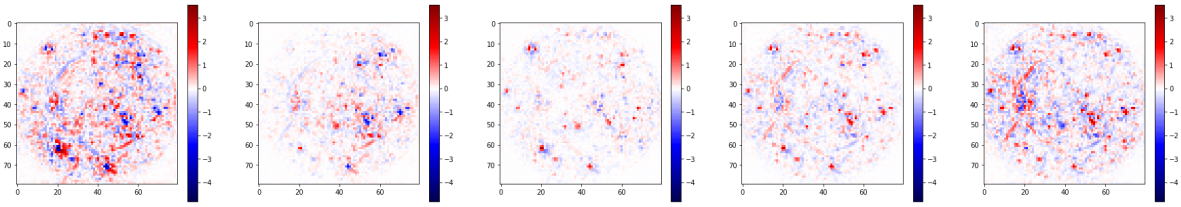
In [66]:

```
vis.plot_scores(img_target,score_rf29)
```



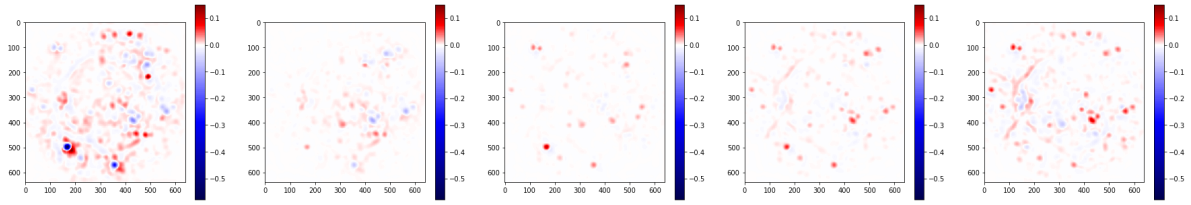
In [67]:

```
#vis.plot_scores(img_target,score_rf29.sum(1).unsqueeze(1), figsize=(50,50), threshold=None
vis.plot_scores(0,score_rf29.sum(1).unsqueeze(0), figsize=(50,50))
```



In [68]:

```
tmp = vis.map_scores_to_input_sz(score_rf29.sum(1).unsqueeze(1), 29)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [69]:

```
#del score_rf637, score_rf509, score_rf381, score_rf253, score_rf189
#del score_rf125, score_rf93, score_rf61, score_rf45, score_mp
#del rf637, rf509, rf381, rf253, rf189, rf125, rf93, rf61, rf45
#del rf509_p, rf253_p, rf125_p, rf61_p
```

In [70]:

```
score_mp = vis.propagate_score_through_maxpool_sz2x2_st2x2(score_rf29, mp_idx)
#del score_rf29, mp_idx
layer = 5
rf21 = layer_vals[layer]
score_rf21, score_k29 = vis.propagate_score_through_conv2d_sz3_pad1(score_mp, rf21, rf29_p,
#del score_mp, rf45
vis.stats(score_rf21)
```

```
Max: 16.687583923339844, Min: -21.55489730834961, Avg: 6.940219486064837e-0
7, Std:0.03120900725217724
torch.Size([5, 64, 160, 160])
```

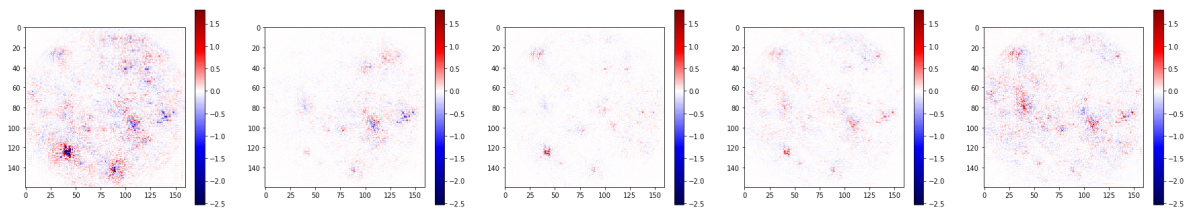
Out[70]:

```
(16.687583923339844,
 -21.55489730834961,
 6.940219486064837e-07,
 0.03120900725217724)
```

In [71]:

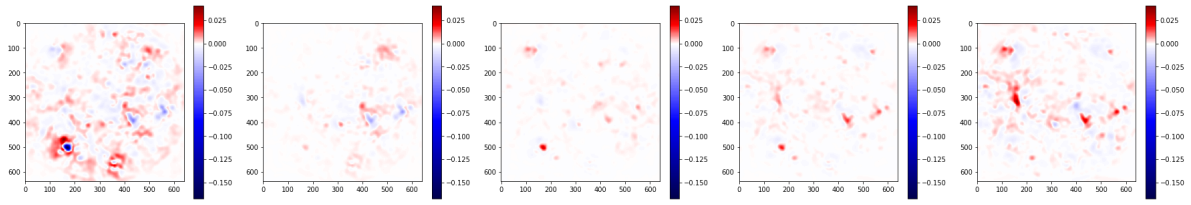
```
score_k29 = score_k29.sum(1).unsqueeze(1)
print(score_k29.size())
#vis.plot_scores(img_target, score_k29, figsize=(50,50))
vis.plot_scores(0,score_k29.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 160, 160])
```



In [72]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k29, 29)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k29
```



```
Max: 0.04146198183298111, Min: -0.18272458016872406, Avg: -2.648821816951114
e-05, Std:0.002957164768684258
torch.Size([5, 1, 640, 640])
```

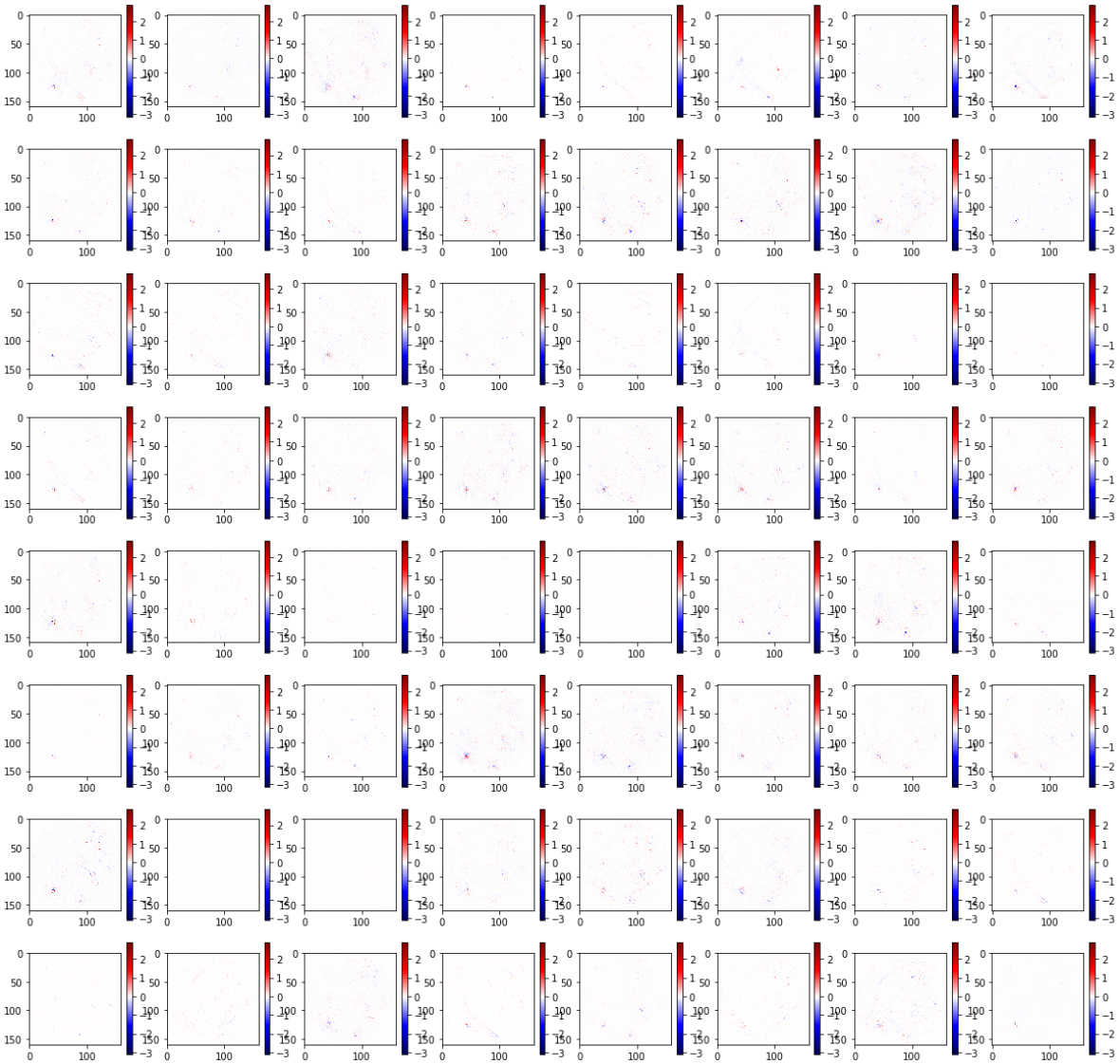
Out[72]:

```
Variable containing:
( 0 , 0 ,.,.) =
  7.3368e-05  7.2653e-05  6.7995e-05  ...   7.4990e-04  7.6378e-04  7.7368
e-04
  7.1799e-05  7.0433e-05  6.4995e-05  ...   7.5215e-04  7.6671e-04  7.7718
e-04
  6.5919e-05  6.3690e-05  5.7390e-05  ...   7.5380e-04  7.6865e-04  7.7944
e-04
                 ...                  ⋱                   ...

  8.4824e-04  8.7353e-04  8.9849e-04  ...   1.9611e-04  1.8863e-04  1.8469
e-04
  8.3507e-04  8.5883e-04  8.8209e-04  ...   2.0545e-04  1.9776e-04  1.9324
e-04
  8.1964e-04  8.4176e-04  8.6328e-04  ...   2.1399e-04  2.0625e-04  2.0128
e-04
      ⋮

( 1 , 0 ,.,.) =
  1.6587e-04  1.7125e-04  1.7541e-04  ...   2.0571e-04  2.1232e-04  2.1783
e-04
  1.7108e-04  1.7693e-04  1.8153e-04  ...   2.0504e-04  2.1197e-04  2.1780
e-04
  1.7453e-04  1.8073e-04  1.8570e-04  ...   2.0474e-04  2.1179e-04  2.1780
e-04
                 ...                  ⋱                   ...

 -5.2970e-05 -5.7933e-05 -6.2857e-05  ...   1.3695e-04  1.3288e-04  1.2903
e-04
 -5.0404e-05 -5.5099e-05 -5.9758e-05  ...   1.3830e-04  1.3429e-04  1.3041
e-04
 -4.7615e-05 -5.2013e-05 -5.6376e-05  ...   1.3861e-04  1.3473e-04  1.3090
e-04
      ⋮

( 2 , 0 ,.,.) =
  2.1783e-04  2.3053e-04  2.4481e-04  ...  -8.3556e-05 -8.4639e-05 -8.5379
e-05
  2.2798e-04  2.4209e-04  2.5782e-04  ...  -8.4237e-05 -8.5383e-05 -8.6168
```

```
e-05
  2.3865e-04  2.5408e-04  2.7110e-04  ...  -8.5035e-05 -8.6198e-05 -8.6991
e-05

                ...                   ⋱                   ...

 -1.2599e-04 -1.3146e-04 -1.3670e-04  ...  -4.2621e-05 -4.2637e-05 -4.3069
e-05
 -1.2275e-04 -1.2785e-04 -1.3274e-04  ...  -4.2847e-05 -4.2852e-05 -4.3247
e-05
 -1.1908e-04 -1.2381e-04 -1.2831e-04  ...  -4.3381e-05 -4.3359e-05 -4.3693
e-05
      ⋮

( 3 , 0 ,.,.) =
 -1.6806e-04 -1.6986e-04 -1.6725e-04  ...  -8.5408e-05 -9.1035e-05 -9.6892
e-05
 -1.6973e-04 -1.7121e-04 -1.6815e-04  ...  -8.2426e-05 -8.8431e-05 -9.4713
e-05
 -1.6640e-04 -1.6732e-04 -1.6374e-04  ...  -8.0979e-05 -8.7164e-05 -9.3681
e-05
                ...                   ⋱                   ...

 -1.2421e-04 -1.2518e-04 -1.2606e-04  ...  -1.2587e-04 -1.2136e-04 -1.1671
e-04
 -1.2223e-04 -1.2308e-04 -1.2382e-04  ...  -1.2373e-04 -1.1953e-04 -1.1514
e-04
 -1.2006e-04 -1.2077e-04 -1.2137e-04  ...  -1.2092e-04 -1.1704e-04 -1.1293
e-04
      ⋮

( 4 , 0 ,.,.) =
 -3.9414e-04 -3.9280e-04 -3.8117e-04  ...  -1.0661e-04 -1.1780e-04 -1.2966
e-04
 -3.9638e-04 -3.9414e-04 -3.8138e-04  ...  -9.8487e-05 -1.1041e-04 -1.2316
e-04
 -3.8787e-04 -3.8437e-04 -3.7047e-04  ...  -9.2890e-05 -1.0521e-04 -1.1851
e-04
                ...                   ⋱                   ...

 -7.5664e-05 -6.7952e-05 -5.9954e-05  ...  -7.1969e-06 -2.7910e-06 -2.3468
e-06
 -7.6837e-05 -6.9383e-05 -6.1611e-05  ...  -1.3379e-05 -9.1331e-06 -8.6497
e-06
 -7.7971e-05 -7.0823e-05 -6.3350e-05  ...  -2.1221e-05 -1.7184e-05 -1.6651
e-05
[torch.FloatTensor of size 5x1x640x640]
```

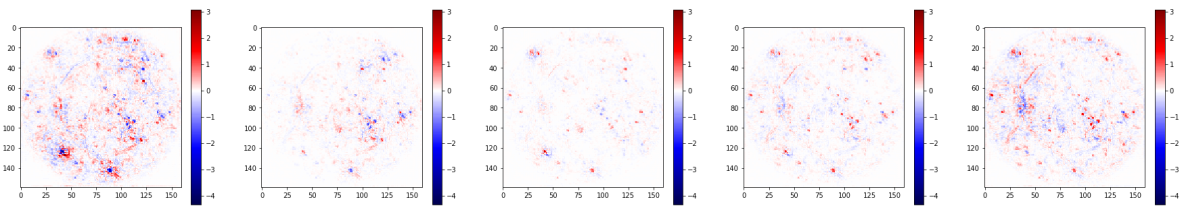
In [73]:

```
vis.plot_scores(img_target, score_rf21)
```



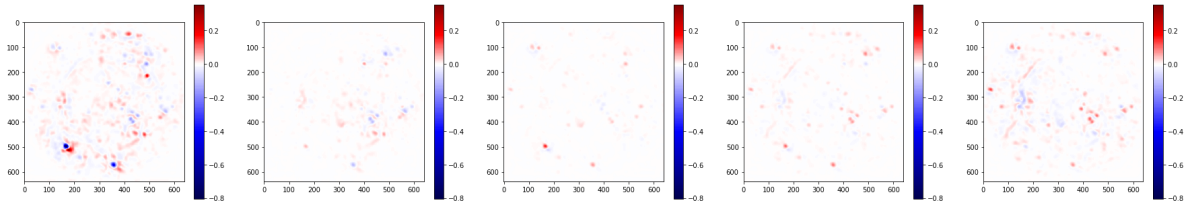
In [74]:

```
#vis.plot_scores(img_target,score_rf21.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf21.sum(1).unsqueeze(0), figsize=(50,50))
```



In [75]:

```
tmp = vis.map_scores_to_input_sz(score_rf21.sum(1).unsqueeze(1), 21)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [76]:

```
layer = 4
rf13_p = layer_vals[layer]
rf13, mp_idx = F.max_pool2d(rf13_p, 2, 2, return_indices = True)  # 1x64x5x5
score_rf13, score_k21 = vis.propagate_score_through_conv2d_sz3_pad1(score_rf21, rf13, rf21,
vis.stats(score_rf13)
```

```
Max: 26.19158935546875, Min: -56.68022918701172, Avg: -1.5063522884623732e-0
5, Std:0.08795524810760041
torch.Size([5, 32, 160, 160])
```

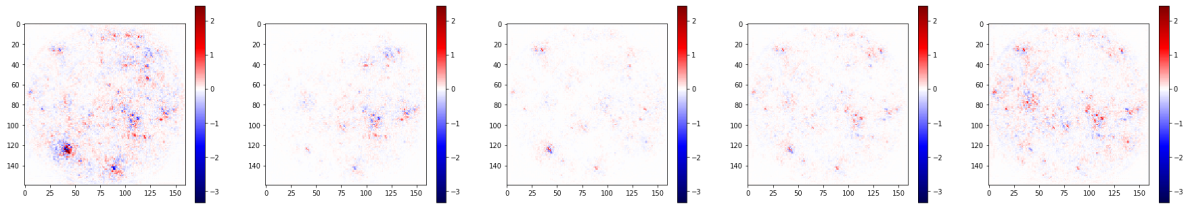
Out[76]:

```
(26.19158935546875,
 -56.68022918701172,
 -1.5063522884623732e-05,
 0.08795524810760041)
```

In [77]:

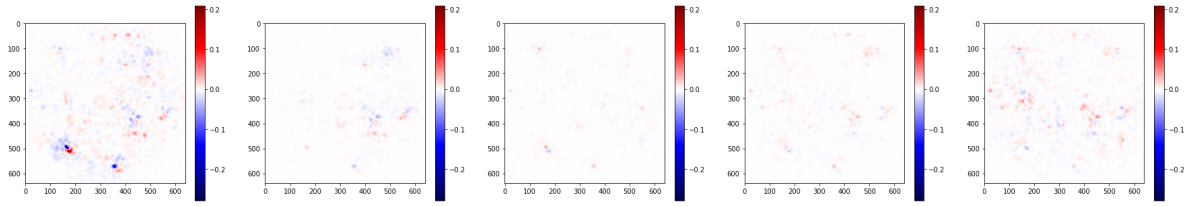
```
score_k21 = score_k21.sum(1).unsqueeze(1)
print(score_k21.size())
#vis.plot_scores(img_target, score_k21, figsize=(50,50))
vis.plot_scores(0,score_k21.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 160, 160])
```



In [78]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k21, 21)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k21
```



```
Max: 0.23147784173488617, Min: -0.32203203439712524, Avg: 3.290313228266869e
-05, Std:0.004440351731681887
torch.Size([5, 1, 640, 640])
```

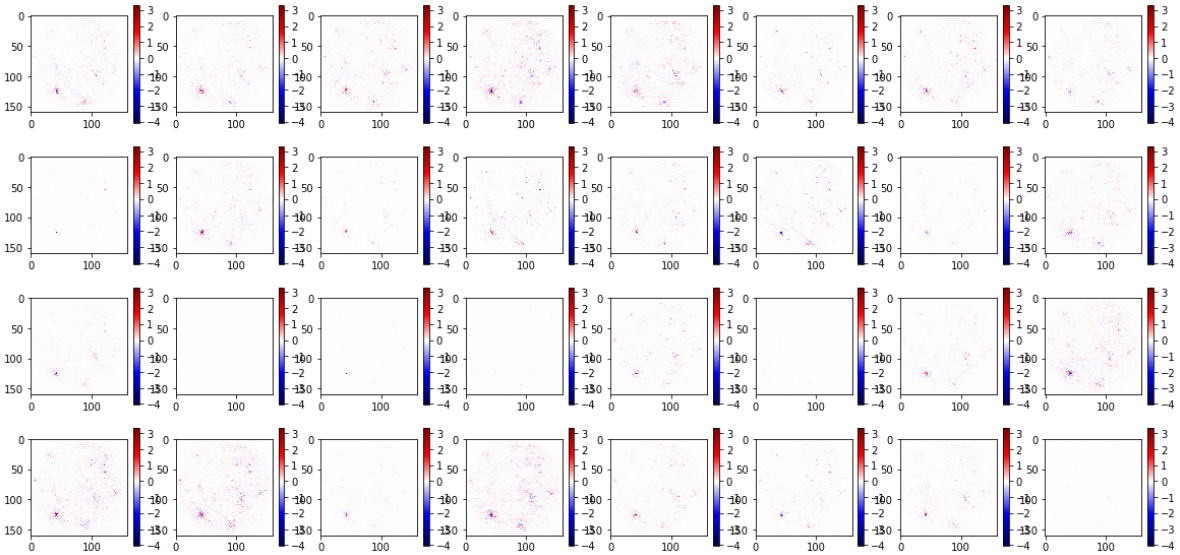
Out[78]:

```
Variable containing:
( 0 , 0 ,.,.) =
  1.1806e-04  1.2089e-04  1.1743e-04  ...   7.5191e-04  7.6221e-04  7.7019
e-04
  1.1950e-04  1.2201e-04  1.1797e-04  ...   7.9530e-04  8.0191e-04  8.0532
e-04
  1.1376e-04  1.1554e-04  1.1082e-04  ...   8.5295e-04  8.5421e-04  8.5109
e-04
                 ...                  ⋱                  ...

  1.0253e-03  1.0757e-03  1.1189e-03  ...   5.1784e-04  4.9743e-04  4.6619
e-04
  1.0079e-03  1.0570e-03  1.0993e-03  ...   4.8795e-04  4.7066e-04  4.4322
e-04
  9.7929e-04  1.0253e-03  1.0654e-03  ...   4.5298e-04  4.3846e-04  4.1494
e-04
      ⋮

( 1 , 0 ,.,.) =
  1.6550e-04  1.6859e-04  1.6957e-04  ...   2.1756e-04  2.2269e-04  2.2671
e-04
  1.6743e-04  1.7010e-04  1.7057e-04  ...   2.3229e-04  2.3602e-04  2.3840
e-04
  1.6616e-04  1.6810e-04  1.6784e-04  ...   2.5209e-04  2.5374e-04  2.5371
e-04
                 ...                  ⋱                  ...

 -1.2165e-04 -1.3714e-04 -1.5058e-04  ...   2.9036e-04  2.7825e-04  2.6025
e-04
 -1.1470e-04 -1.2925e-04 -1.4189e-04  ...   2.7731e-04  2.6635e-04  2.4986
e-04
 -1.0516e-04 -1.1839e-04 -1.2991e-04  ...   2.5955e-04  2.4988e-04  2.3523
e-04
      ⋮

( 2 , 0 ,.,.) =
  1.8047e-04  1.8716e-04  1.9622e-04  ...  -8.0645e-05 -8.1279e-05 -8.1954
e-05
  1.8398e-04  1.9084e-04  2.0017e-04  ...  -8.2548e-05 -8.3030e-05 -8.3547
```

```
e-05
  1.8867e-04  1.9561e-04  2.0504e-04  ...  -8.5246e-05 -8.5489e-05 -8.5745
e-05

                ...                   ⋱                   ...

 -1.5911e-04 -1.6819e-04 -1.7539e-04  ...  -2.0446e-05 -2.2717e-05 -2.5854
e-05
 -1.5488e-04 -1.6367e-04 -1.7071e-04  ...  -2.1395e-05 -2.3664e-05 -2.6727
e-05
 -1.4863e-04 -1.5687e-04 -1.6355e-04  ...  -2.3689e-05 -2.5801e-05 -2.8619
e-05
      ⋮

( 3 , 0 ,.,.) =
 -1.9213e-04 -1.9410e-04 -1.8974e-04  ...  -9.2677e-05 -9.7607e-05 -1.0259
e-04
 -1.9123e-04 -1.9212e-04 -1.8651e-04  ...  -9.3540e-05 -9.8230e-05 -1.0303
e-04
 -1.8194e-04 -1.8118e-04 -1.7404e-04  ...  -9.6680e-05 -1.0077e-04 -1.0504
e-04
                ...                   ⋱                   ...

  2.5194e-05  4.8924e-05  6.9295e-05  ...  -1.9289e-04 -1.8763e-04 -1.7853
e-04
  1.7634e-05  3.9830e-05  5.8858e-05  ...  -1.8561e-04 -1.8100e-04 -1.7267
e-04
  5.1575e-06  2.5026e-05  4.2045e-05  ...  -1.7557e-04 -1.7154e-04 -1.6409
e-04
      ⋮

( 4 , 0 ,.,.) =
 -4.1027e-04 -4.0517e-04 -3.8716e-04  ...  -1.4059e-04 -1.5057e-04 -1.5963
e-04
 -4.0203e-04 -3.9351e-04 -3.7180e-04  ...  -1.4894e-04 -1.5779e-04 -1.6559
e-04
 -3.7666e-04 -3.6357e-04 -3.3762e-04  ...  -1.6298e-04 -1.6978e-04 -1.7544
e-04
                ...                   ⋱                   ...

  3.1260e-04  3.8125e-04  4.3967e-04  ...  -1.3527e-04 -1.3035e-04 -1.2178
e-04
  2.8961e-04  3.5437e-04  4.0944e-04  ...  -1.3079e-04 -1.2669e-04 -1.1914
e-04
  2.5220e-04  3.1085e-04  3.6076e-04  ...  -1.2436e-04 -1.2095e-04 -1.1449
e-04
[torch.FloatTensor of size 5x1x640x640]
```

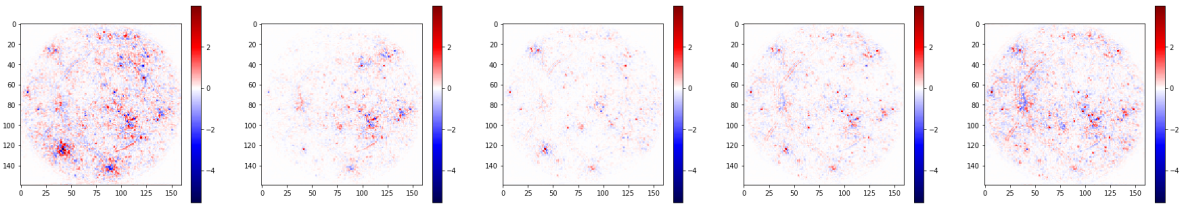
In [79]:

```
vis.plot_scores(img_target, score_rf13)
```



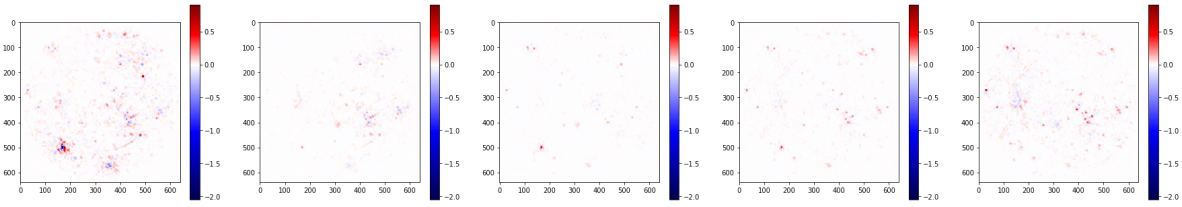
In [80]:

```
#vis.plot_scores(img_target,score_rf13.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf13.sum(1).unsqueeze(0), figsize=(50,50))
```



In [81]:

```
tmp = vis.map_scores_to_input_sz(score_rf13.sum(1).unsqueeze(1), 13)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [82]:

```
score_mp = vis.propagate_score_through_maxpool_sz2x2_st2x2(score_rf13, mp_idx)
#del score_rf29, mp_idx
layer = 3
rf9 = layer_vals[layer]
score_rf9, score_k13 = vis.propagate_score_through_conv2d_sz3_pad1(score_mp, rf9, rf13_p, m
#del score_mp, rf45
vis.stats(score_rf9)
```

```
Max: 14.683265686035156, Min: -31.207115173339844, Avg: -7.416107391821525e-
06, Std:0.031944198540474256
torch.Size([5, 32, 320, 320])
```

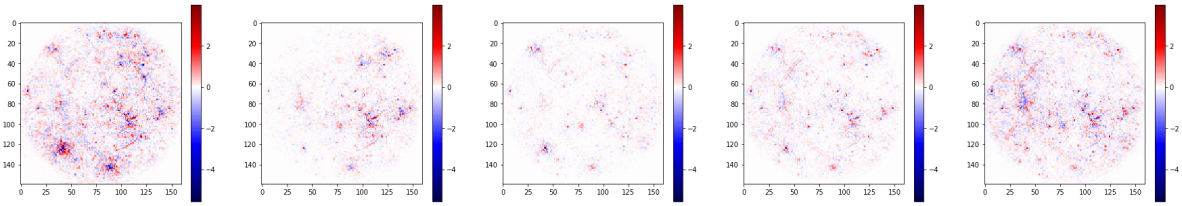
Out[82]:

```
(14.683265686035156,
 -31.207115173339844,
 -7.416107391821525e-06,
 0.031944198540474256)
```

In [83]:

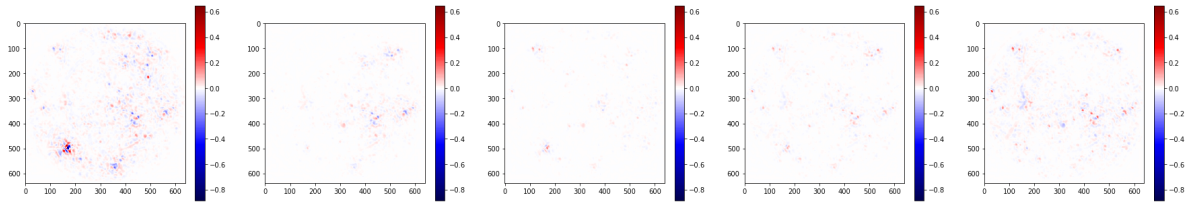
```
score_k13 = score_k13.sum(1).unsqueeze(1)
print(score_k13.size())
#vis.plot_scores(img_target, score_k13, figsize=(50,50))
vis.plot_scores(0,score_rf13.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 320, 320])
```



In [84]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k13, 13)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k13
```



```
Max: 0.9088202118873596, Min: -1.4287272691726685, Avg: 2.9201817954646324e-
05, Std:0.01387330661832968
torch.Size([5, 1, 640, 640])
```

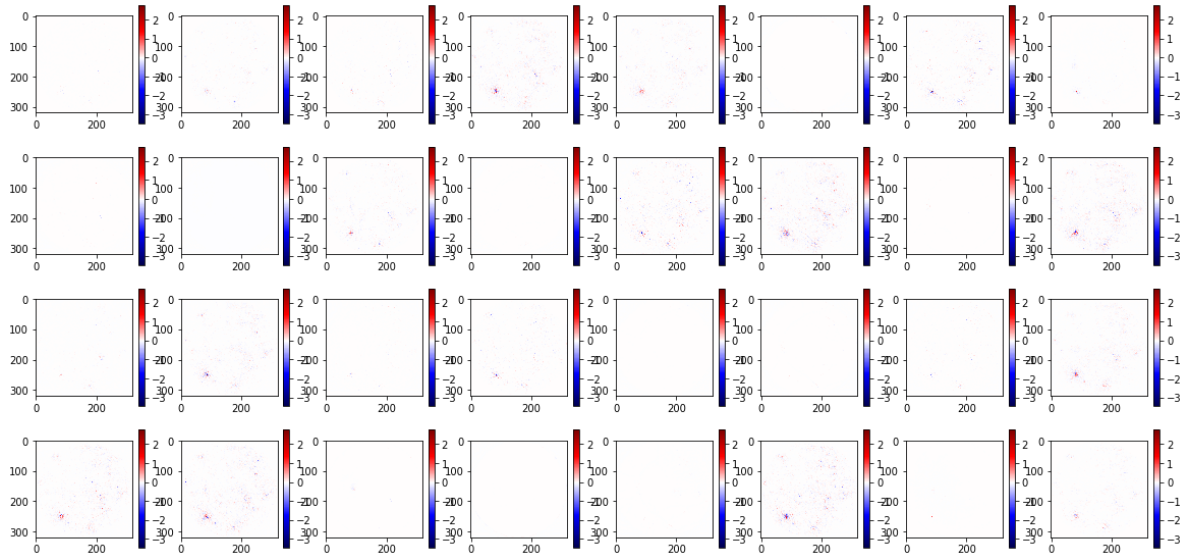
Out[84]:

```
Variable containing:
( 0 , 0 ,.,.) =
  3.8297e-04  4.1099e-04  4.0081e-04  ...   1.8023e-03  1.6064e-03  1.3251
e-03
  4.7172e-04  5.1479e-04  5.1004e-04  ...   2.3045e-03  2.0186e-03  1.6205
e-03
  5.3932e-04  6.0247e-04  6.1178e-04  ...   2.7997e-03  2.4240e-03  1.9160
e-03
                 ...                  ⋱                   ...

  1.5245e-03  1.7156e-03  1.8713e-03  ...   1.5168e-03  1.5590e-03  1.4510
e-03
  1.4017e-03  1.5594e-03  1.6881e-03  ...   1.3791e-03  1.4357e-03  1.3514
e-03
  1.2546e-03  1.3762e-03  1.4765e-03  ...   1.1604e-03  1.2220e-03  1.1650
e-03
      ⋮

( 1 , 0 ,.,.) =
  1.5185e-04  1.5264e-04  1.5304e-04  ...   5.8002e-04  5.1293e-04  4.1681
e-04
  1.3896e-04  1.3628e-04  1.3368e-04  ...   7.5447e-04  6.5622e-04  5.1952
e-04
  1.2007e-04  1.1299e-04  1.0659e-04  ...   9.2699e-04  7.9773e-04  6.2278
e-04
                 ...                  ⋱                   ...

 -2.0411e-04 -2.5239e-04 -2.9535e-04  ...   6.8644e-04  7.1191e-04  6.7281
e-04
 -1.8431e-04 -2.2644e-04 -2.6391e-04  ...   6.2416e-04  6.5401e-04  6.2413
e-04
 -1.5744e-04 -1.9211e-04 -2.2317e-04  ...   5.3089e-04  5.6033e-04  5.4003
e-04
      ⋮

( 2 , 0 ,.,.) =
 -4.8328e-05 -5.9652e-05 -4.1441e-05  ...  -1.4524e-04 -1.3328e-04 -1.1644
e-04
 -1.3937e-04 -1.6649e-04 -1.5511e-04  ...  -1.7639e-04 -1.5887e-04 -1.3483
```

```
e-04
 -2.2229e-04 -2.7077e-04 -2.7425e-04  ...  -2.0727e-04 -1.8412e-04 -1.5322
e-04

                ...                   ⋱                ...

 -2.2939e-04 -2.5728e-04 -2.7859e-04  ...   1.7051e-05  1.7028e-05  1.1361
e-05
 -2.1200e-04 -2.3537e-04 -2.5307e-04  ...   6.5683e-06  6.7733e-06  2.3802
e-06
 -1.9024e-04 -2.0870e-04 -2.2270e-04  ...  -4.9230e-06 -4.8064e-06 -8.1219
e-06
      ⋮

( 3 , 0 ,.,.) =
 -3.6045e-04 -3.7131e-04 -3.5621e-04  ...  -1.9110e-04 -1.7869e-04 -1.5801
e-04
 -4.1707e-04 -4.3528e-04 -4.2035e-04  ...  -2.3176e-04 -2.1322e-04 -1.8349
e-04
 -4.5668e-04 -4.8654e-04 -4.7836e-04  ...  -2.7198e-04 -2.4770e-04 -2.0976
e-04
                ...                   ⋱                ...

  2.2725e-04  3.1974e-04  3.9783e-04  ...  -3.9974e-04 -4.2383e-04 -4.1031
e-04
  1.8083e-04  2.5938e-04  3.2615e-04  ...  -3.7210e-04 -3.9873e-04 -3.8956
e-04
  1.2221e-04  1.8450e-04  2.3832e-04  ...  -3.2482e-04 -3.5017e-04 -3.4502
e-04
      ⋮

( 4 , 0 ,.,.) =
 -7.0877e-04 -6.9611e-04 -6.3698e-04  ...  -4.0483e-04 -3.6467e-04 -3.0174
e-04
 -8.2067e-04 -8.1637e-04 -7.4978e-04  ...  -5.2129e-04 -4.6203e-04 -3.7243
e-04
 -9.0520e-04 -9.2088e-04 -8.6178e-04  ...  -6.3646e-04 -5.5914e-04 -4.4517
e-04
                ...                   ⋱                ...

  8.5038e-04  1.0988e-03  1.3064e-03  ...  -1.0159e-03 -1.0035e-03 -8.9105
e-04
  7.3391e-04  9.4762e-04  1.1269e-03  ...  -8.8130e-04 -8.9094e-04 -8.0760
e-04
  5.7916e-04  7.5124e-04  8.9758e-04  ...  -7.0061e-04 -7.2479e-04 -6.7121
e-04
[torch.FloatTensor of size 5x1x640x640]
```

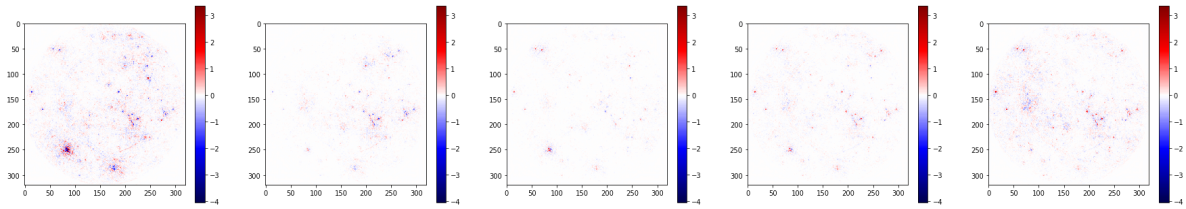
In [85]:

```
vis.plot_scores(img_target, score_rf9)
```



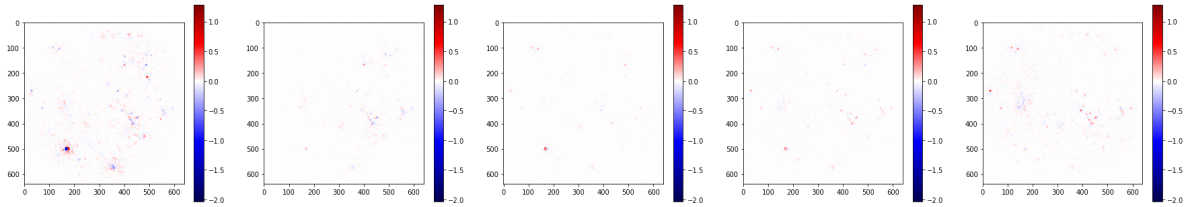
In [86]:

```
#vis.plot_scores(img_target,score_rf9.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf9.sum(1).unsqueeze(0), figsize=(50,50))
```



In [87]:

```
tmp = vis.map_scores_to_input_sz(score_rf9.sum(1).unsqueeze(1), 9)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [88]:

```
layer = 2
rf5_p = layer_vals[layer]
rf5, mp_idx = F.max_pool2d(rf5_p, 2, 2, return_indices = True)  # 1x64x5x5
score_rf5, score_k9 = vis.propagate_score_through_conv2d_sz3_pad1(score_rf9, rf5, rf9, me.r
vis.stats(score_rf5)
```

```
Max: 113.83822631835938, Min: -146.29376220703125, Avg: 1.1647287478473014e-
05, Std:0.15557174397678317
torch.Size([5, 16, 320, 320])
```

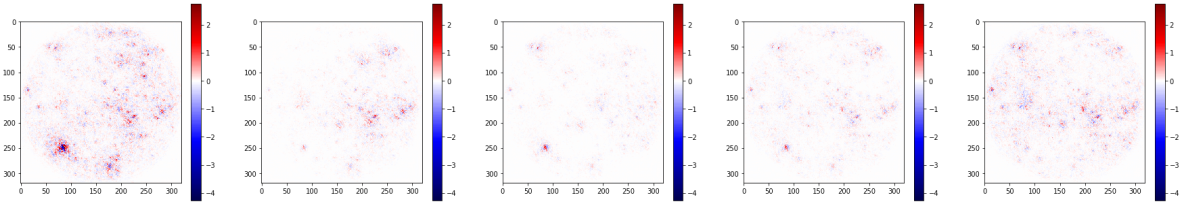
Out[88]:

```
(113.83822631835938,
 -146.29376220703125,
 1.1647287478473014e-05,
 0.15557174397678317)
```

In [89]:

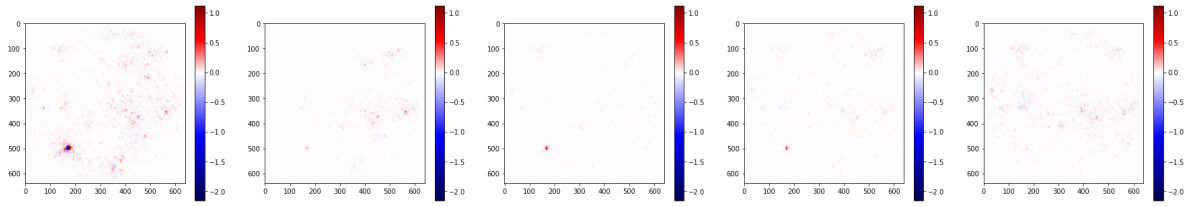
```
score_k9 = score_k9.sum(1).unsqueeze(1)
print(score_k9.size())
#vis.plot_scores(img_target, score_k9, figsize=(50,50))
vis.plot_scores(0,score_k9.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 320, 320])
```



In [90]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k9, 9)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k9
```



```
Max: 2.0563790798187256, Min: -7.680121421813965, Avg: -0.000105918006713223
58, Std:0.03674478537726132
torch.Size([5, 1, 640, 640])
```

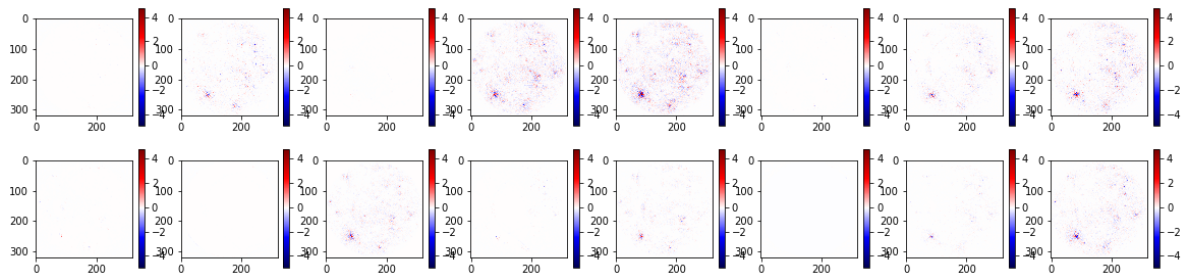
Out[90]:

```
Variable containing:
( 0 , 0 ,.,.) =
  9.6622e-04  1.2228e-03  1.3375e-03  ...   3.6922e-03  3.1976e-03  2.4405
e-03
  1.2177e-03  1.5375e-03  1.6713e-03  ...   5.1991e-03  4.3430e-03  3.1689
e-03
  1.3167e-03  1.6488e-03  1.7761e-03  ...   6.4671e-03  5.2469e-03  3.7044
e-03
                 ...                  ⋱                  ...

  2.8575e-03  3.6622e-03  4.2306e-03  ...   5.6263e-03  5.5171e-03  4.5695
e-03
  2.5234e-03  3.1556e-03  3.5755e-03  ...   5.2666e-03  4.9353e-03  3.9557
e-03
  2.0386e-03  2.4621e-03  2.7258e-03  ...   4.1789e-03  3.7846e-03  2.9687
e-03
      ⋮

( 1 , 0 ,.,.) =
  2.6499e-04  3.0003e-04  3.1102e-04  ...   1.2603e-03  1.0901e-03  8.2439
e-04
  2.4014e-04  2.5789e-04  2.5102e-04  ...   1.7634e-03  1.4757e-03  1.0722
e-03
  1.6818e-04  1.5179e-04  1.1780e-04  ...   2.1690e-03  1.7656e-03  1.2451
e-03
                 ...                  ⋱                  ...

 -4.1701e-04 -5.6143e-04 -6.6852e-04  ...   2.3672e-03  2.3314e-03  1.9490
e-03
 -3.6425e-04 -4.9094e-04 -5.8725e-04  ...   2.2312e-03  2.1030e-03  1.7040
e-03
 -2.8384e-04 -3.8023e-04 -4.5587e-04  ...   1.7895e-03  1.6331e-03  1.2981
e-03
      ⋮

( 2 , 0 ,.,.) =
 -2.8655e-04 -4.0636e-04 -4.5914e-04  ...  -2.4617e-04 -2.1777e-04 -1.7533
e-04
 -5.1043e-04 -6.9730e-04 -7.8398e-04  ...  -3.4137e-04 -2.8917e-04 -2.2001
```

```
e-04
 -6.9573e-04 -9.3873e-04 -1.0546e-03  ...   -4.2774e-04 -3.5032e-04 -2.5579
e-04

                ...                   ⋱                ...

 -4.0261e-04 -5.1182e-04 -5.8903e-04  ...    1.5237e-04  1.3897e-04  1.0224
e-04
 -3.6329e-04 -4.5254e-04 -5.1221e-04  ...    1.2598e-04  1.1221e-04  7.9511
e-05
 -2.9986e-04 -3.6254e-04 -4.0224e-04  ...    8.2491e-05  7.0893e-05  4.6251
e-05
      ⋮

( 3 , 0 ,.,.) =
 -7.9591e-04 -9.7638e-04 -1.0528e-03  ...   -4.7637e-04 -4.2759e-04 -3.3835
e-04
 -9.4538e-04 -1.1528e-03 -1.2262e-03  ...   -6.0552e-04 -5.3447e-04 -4.1288
e-04
 -9.6857e-04 -1.1596e-03 -1.2070e-03  ...   -6.7642e-04 -5.8907e-04 -4.4912
e-04
                ...                   ⋱                ...

  8.1893e-04  1.1556e-03  1.3794e-03  ...   -1.3401e-03 -1.3340e-03 -1.1299
e-03
  6.8420e-04  9.6467e-04  1.1478e-03  ...   -1.3082e-03 -1.2383e-03 -1.0123
e-03
  4.7808e-04  6.7899e-04  8.0969e-04  ...   -1.0809e-03 -9.8848e-04 -7.9194
e-04
      ⋮

( 4 , 0 ,.,.) =
 -1.3793e-03 -1.6350e-03 -1.7250e-03  ...   -1.1300e-03 -9.9442e-04 -7.5617
e-04
 -1.6250e-03 -1.9037e-03 -1.9632e-03  ...   -1.4652e-03 -1.2687e-03 -9.4532
e-04
 -1.6688e-03 -1.9005e-03 -1.8893e-03  ...   -1.6496e-03 -1.4080e-03 -1.0360
e-03
                ...                   ⋱                ...

  2.4463e-03  3.3228e-03  3.8821e-03  ...   -4.2847e-03 -3.9396e-03 -3.0740
e-03
  2.1246e-03  2.8714e-03  3.3390e-03  ...   -3.8498e-03 -3.4446e-03 -2.6307
e-03
  1.5855e-03  2.1335e-03  2.4757e-03  ...   -2.9286e-03 -2.5670e-03 -1.9343
e-03
[torch.FloatTensor of size 5x1x640x640]
```

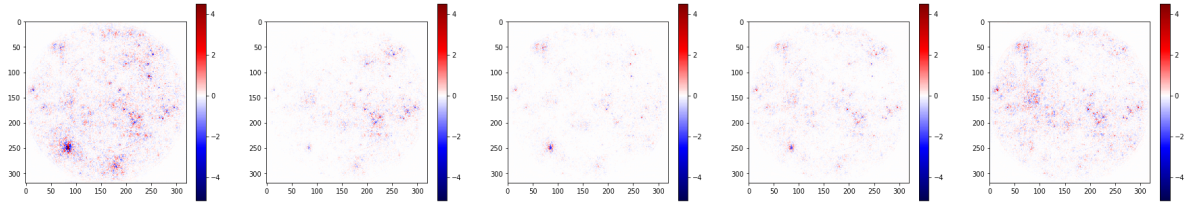
In [91]:

```
vis.plot_scores(img_target, score_rf5)
```



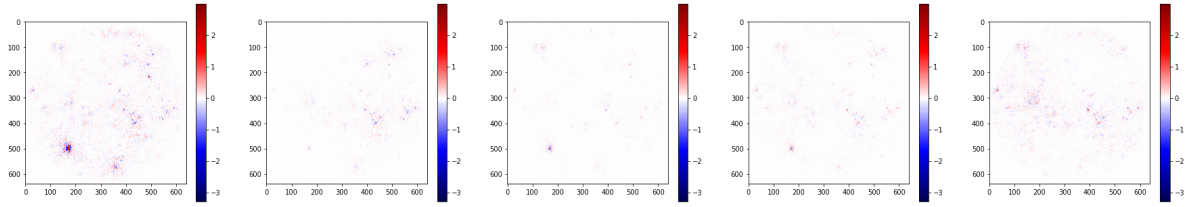
In [92]:

```
#vis.plot_scores(img_target,score_rf5.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf5.sum(1).unsqueeze(0), figsize=(50,50))
```



In [93]:

```
tmp = vis.map_scores_to_input_sz(score_rf5.sum(1).unsqueeze(1), 5)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [94]:

```
#del score_rf21, score_rf13, score_rf9
#del rf21, rf13, rf9
#del rf29_p, rf13_p
```

In [95]:

```
score_mp = vis.propagate_score_through_maxpool_sz2x2_st2x2(score_rf5, mp_idx)
#del score_rf29, mp_idx
layer = 1
rf3 = layer_vals[layer]
score_rf3, score_k5 = vis.propagate_score_through_conv2d_sz3_pad1(score_mp, rf3, rf5_p, me.
#del score_mp, rf4
vis.stats(score_rf3)
```

```
Max: 26.007694244384766, Min: -109.79852294921875, Avg: 3.4425979869134966e-
07, Std:0.06383974925232758
torch.Size([5, 16, 640, 640])
```

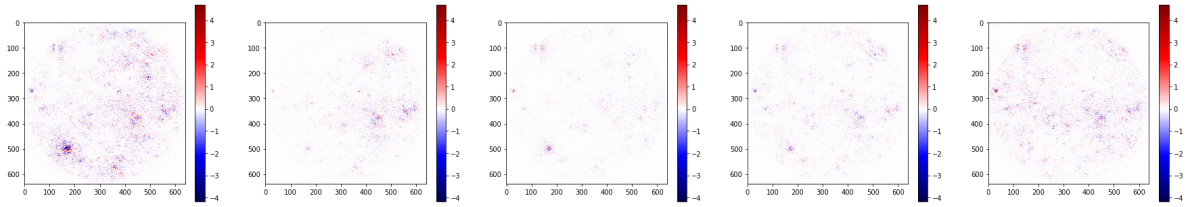
Out[95]:

```
(26.007694244384766,
 -109.79852294921875,
 3.4425979869134966e-07,
 0.06383974925232758)
```

In [96]:

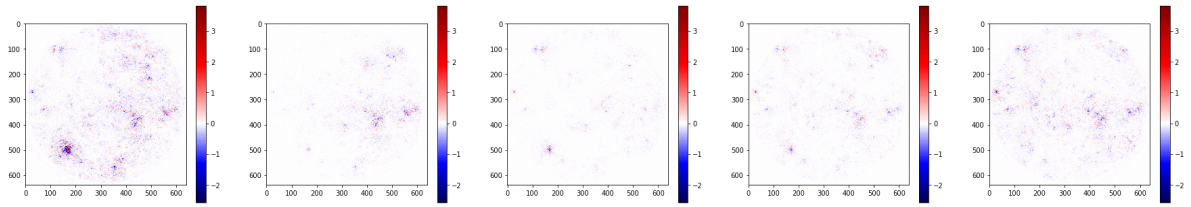
```
score_k5 = score_k5.sum(1).unsqueeze(1)
print(score_k5.size())
#vis.plot_scores(img_target, score_k5, figsize=(50,50))
vis.plot_scores(0,score_k5.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 640, 640])
```



In [97]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k5, 5)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k5
```



```
Max: 44.11140060424805, Min: -12.16321086883545, Avg: 4.1080999517524865e-0
5, Std:0.15175292834010756
torch.Size([5, 1, 640, 640])
```

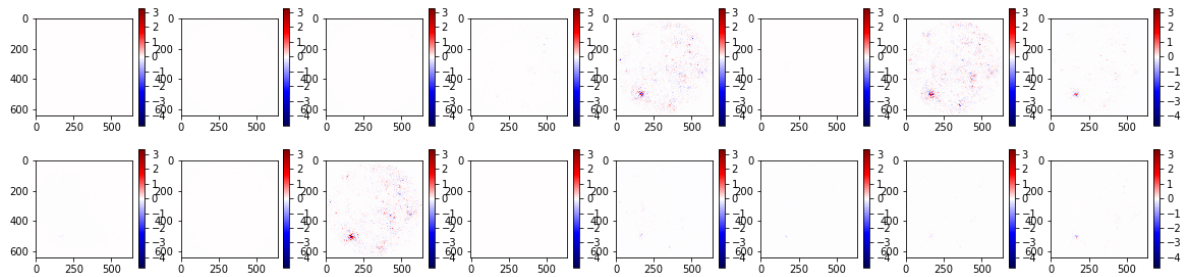
Out[97]:

```
Variable containing:
( 0 , 0 ,.,.) =
 -2.4354e-03  1.3259e-03  1.8352e-03  ...   2.8854e-03  1.8272e-03  1.9070
e-03
  2.6266e-03  5.1312e-03  4.1485e-03  ...   4.2896e-03  3.8679e-03  3.1096
e-03
  4.0753e-03  5.0905e-03  3.6341e-03  ...   4.8823e-03  4.9781e-03  4.0058
e-03
                 ...                   ⋱                   ...

  3.7181e-03  6.3010e-03  6.3132e-03  ...   5.9503e-03  6.8410e-03  5.6982
e-03
  4.3006e-03  7.7393e-03  7.3759e-03  ...   6.8868e-03  1.0089e-02  8.9931
e-03
  3.7423e-03  6.0821e-03  5.8153e-03  ...   5.1387e-03  8.2432e-03  7.6501
e-03
      ⋮

( 1 , 0 ,.,.) =
 -3.5253e-04  5.1382e-04  6.4267e-04  ...   1.0426e-03  6.7087e-04  6.6675
e-04
  1.1601e-03  1.7679e-03  1.6012e-03  ...   1.5913e-03  1.4534e-03  1.1153
e-03
  1.4768e-03  1.8414e-03  1.5736e-03  ...   1.7639e-03  1.8025e-03  1.3853
e-03
                 ...                   ⋱                   ...

 -6.0421e-04 -9.1000e-04 -9.1951e-04  ...   2.5559e-03  2.9150e-03  2.4124
e-03
 -5.0503e-04 -7.8570e-04 -8.0321e-04  ...   2.9189e-03  4.2326e-03  3.7491
e-03
 -3.3805e-04 -4.6971e-04 -5.0875e-04  ...   2.1843e-03  3.4574e-03  3.1973
e-03
      ⋮

( 2 , 0 ,.,.) =
  1.5720e-03 -2.7227e-04 -5.1507e-04  ...  -1.7941e-04 -1.0691e-04 -1.1980
e-04
 -1.9717e-04 -1.3706e-03 -8.3226e-04  ...  -2.2865e-04 -1.7286e-04 -1.4477
```

```
e-04
 -9.4956e-04 -1.2850e-03 -4.9220e-04  ...  -2.6344e-04 -2.4430e-04 -2.0654
e-04

                 ...                   ⋱                   ...

 -5.1499e-04 -8.7393e-04 -8.7127e-04  ...   2.0018e-04  1.9103e-04  1.2855
e-04
 -5.9783e-04 -1.0619e-03 -1.0045e-03  ...   1.8126e-04  2.4338e-04  1.9430
e-04
 -5.2531e-04 -8.3146e-04 -7.8913e-04  ...   1.0885e-04  1.7975e-04  1.5629
e-04
      ⋮

( 3 , 0 ,.,.) =
  2.3699e-03 -1.0859e-03 -1.5845e-03  ...  -5.1908e-04 -4.4045e-04 -3.7162
e-04
 -2.5366e-03 -4.6623e-03 -3.7483e-03  ...  -7.5549e-04 -7.8367e-04 -5.9746
e-04
 -3.9277e-03 -4.7240e-03 -3.3221e-03  ...  -7.4080e-04 -8.1577e-04 -6.2857
e-04
                 ...                   ⋱                   ...

  1.1575e-03  2.0403e-03  2.0370e-03  ...  -1.4143e-03 -1.7004e-03 -1.4479
e-03
  1.1273e-03  2.0944e-03  2.0130e-03  ...  -1.7323e-03 -2.5583e-03 -2.2854
e-03
  7.6803e-04  1.3559e-03  1.3422e-03  ...  -1.3508e-03 -2.1201e-03 -1.9614
e-03
      ⋮

( 4 , 0 ,.,.) =
  5.1108e-03 -1.8079e-03 -2.7913e-03  ...  -1.2287e-03 -1.0164e-03 -8.5993
e-04
 -4.6016e-03 -8.5852e-03 -6.6368e-03  ...  -1.9328e-03 -2.0530e-03 -1.5510
e-03
 -7.5627e-03 -8.8662e-03 -5.8186e-03  ...  -1.9455e-03 -2.1753e-03 -1.6352
e-03
                 ...                   ⋱                   ...

  3.3121e-03  5.7122e-03  5.6267e-03  ...  -4.7257e-03 -5.0485e-03 -4.0691
e-03
  3.3807e-03  6.0708e-03  5.7622e-03  ...  -5.2434e-03 -7.1546e-03 -6.2882
e-03
  2.4332e-03  4.0870e-03  3.9886e-03  ...  -3.8917e-03 -5.8225e-03 -5.3361
e-03
[torch.FloatTensor of size 5x1x640x640]
```

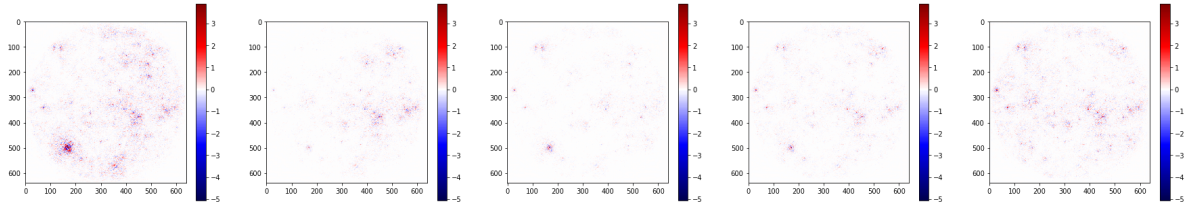
In [98]:

```
vis.plot_scores(img_target, score_rf3)
```



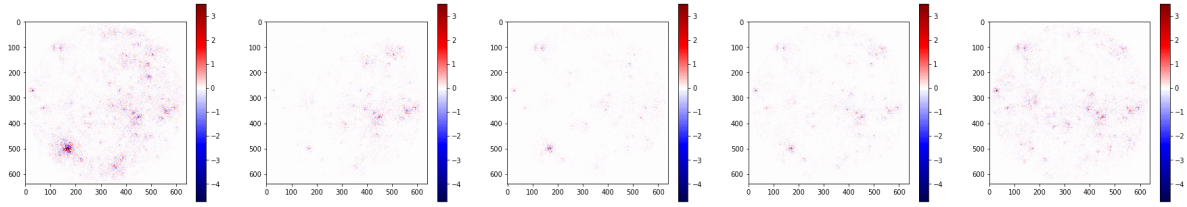
In [99]:

```
#vis.plot_scores(img_target,score_rf3.sum(1).unsqueeze(1), figsize=(50,50))
vis.plot_scores(0,score_rf3.sum(1).unsqueeze(0), figsize=(50,50))
```



In [100]:

```
tmp = vis.map_scores_to_input_sz(score_rf3.sum(1).unsqueeze(1), 3)
#vis.plot_scores(img_target,tmp, figsize=(50,50))
vis.plot_scores(0,tmp.sum(1).unsqueeze(0), figsize=(50,50))
del tmp
```



In [101]:

```
#del score_rf5, score_mp
#del rf5_p
```

In [102]:

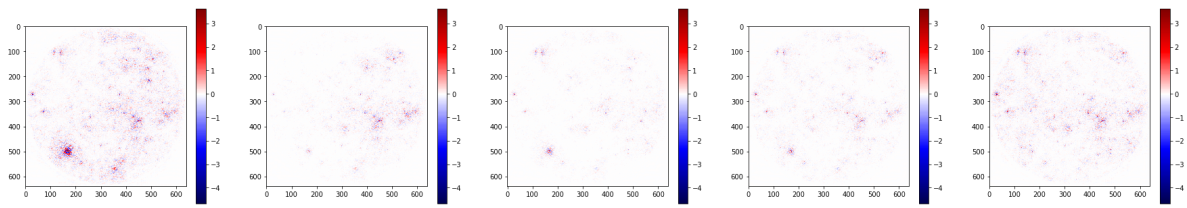
```
layer = 0
score_input, score_k3 = vis.propagate_score_through_conv2d_sz3_pad1(score_rf3, input, rf3,
_, _, _, std = vis.stats(score_input)
```

```
Max: 67.8798828125, Min: -58.36824035644531, Avg: 1.6174451085650687e-06, St
d:0.18160277832651198
torch.Size([5, 3, 640, 640])
```

In [103]:

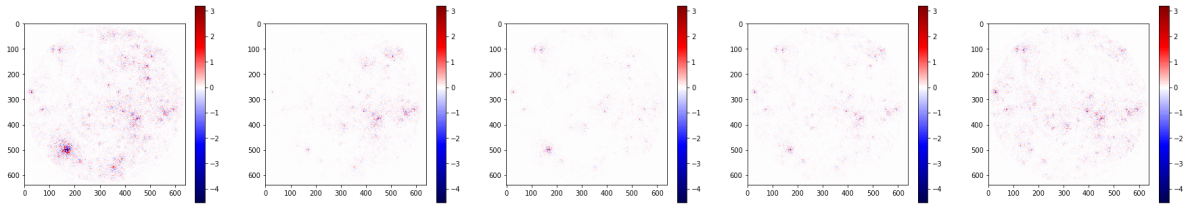
```
score_k3 = score_k3.sum(1).unsqueeze(1)
print(score_k3.size())
#vis.plot_scores(img_target, score_k3, figsize=(50,50))
vis.plot_scores(0,score_k3.sum(1).unsqueeze(0), figsize=(50,50))
```

```
torch.Size([5, 1, 640, 640])
```



In [104]:

```
# map to input space
score_kmapped = vis.map_scores_to_input_sz(score_k3, 3)
#vis.plot_scores(img_target,score_kmapped, figsize=(50,50))
vis.plot_scores(0,score_kmapped.sum(1).unsqueeze(0), figsize=(50,50))
vis.stats(score_kmapped)
score_accumulated_k.add_(score_kmapped)
#del score_k3
```



```
Max: 23.442638397216797, Min: -93.74502563476562, Avg: 6.558044464284735e-0
7, Std:0.1853618693433226
torch.Size([5, 1, 640, 640])
```

Out[104]:

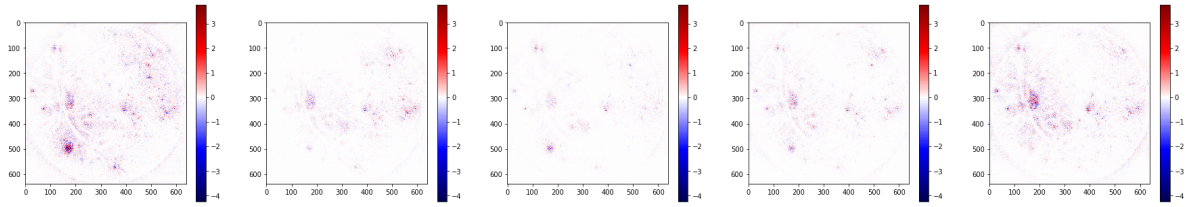
```
Variable containing:
( 0 , 0 ,.,.) =
  1.5588e-03  3.9740e-03  3.6614e-03  ...   3.7961e-03  3.2132e-03  2.7755
e-03
  5.4408e-03  6.3912e-03  6.2892e-03  ...   5.2104e-03  5.3722e-03  4.1610
e-03
  3.1439e-03  3.7419e-03  6.1851e-03  ...   8.8519e-03  7.2632e-03  5.4751
e-03
                 ...                  ⋱                 ...

  1.6072e-03  4.7176e-03  5.1430e-03  ...   5.6968e-03  5.5520e-03  5.5496
e-03
  2.5043e-03  6.3408e-03  6.3587e-03  ...   6.0627e-03  8.7924e-03  8.6929
e-03
  3.5155e-03  6.2064e-03  6.0103e-03  ...   5.2795e-03  8.4906e-03  7.8570
e-03
      ⋮

( 1 , 0 ,.,.) =
 -3.3500e-04  1.7047e-04 -2.0131e-04  ...   1.4010e-03  1.0524e-03  9.1298
e-04
  4.4740e-04  7.6107e-04  6.7469e-04  ...   1.9363e-03  1.9052e-03  1.4466
e-03
 -1.8738e-04  5.0336e-05  4.7245e-04  ...   3.0539e-03  2.4805e-03  1.8329
e-03
                 ...                  ⋱                 ...

 -5.9192e-04 -1.0771e-03 -1.2594e-03  ...   2.3093e-03  2.2854e-03  2.2695
e-03
 -4.6572e-04 -7.8671e-04 -8.5123e-04  ...   2.4949e-03  3.6295e-03  3.5841
e-03
 -3.4841e-04 -4.7878e-04 -5.2011e-04  ...   2.2296e-03  3.5433e-03  3.2713
e-03
      ⋮

( 2 , 0 ,.,.) =
 -1.2723e-03 -2.5956e-03 -2.7557e-03  ...  -2.1744e-04 -2.0143e-04 -1.7612
e-04
 -3.1028e-03 -3.5874e-03 -3.6651e-03  ...  -3.2453e-04 -3.0453e-04 -2.1786
```

```
e-04
 -2.1314e-03 -2.4800e-03 -3.9539e-03  ...  -6.0326e-04 -4.5696e-04 -3.2162
e-04

                ...                   ⋱                   ...

 -2.5782e-04 -6.8397e-04 -7.5306e-04  ...   1.6603e-04  1.2696e-04  9.0674
e-05
 -3.6784e-04 -8.9202e-04 -8.8603e-04  ...   1.2710e-04  1.7861e-04  1.6631
e-04
 -4.9154e-04 -8.4834e-04 -8.1574e-04  ...   1.0718e-04  1.7917e-04  1.5629
e-04
      ⋮

( 3 , 0 ,.,.) =
 -5.6380e-04 -2.8870e-03 -2.5693e-03  ...  -5.6597e-04 -3.5904e-04 -3.4389
e-04
 -4.0140e-03 -4.9662e-03 -5.0123e-03  ...  -8.1730e-04 -8.0484e-04 -6.4901
e-04
 -1.8596e-03 -2.4089e-03 -4.7461e-03  ...  -9.9665e-04 -8.5987e-04 -6.9144
e-04
                ...                   ⋱                   ...

  1.0796e-03  2.1647e-03  2.4357e-03  ...  -1.1902e-03 -1.3140e-03 -1.3726
e-03
  9.4816e-04  1.9030e-03  1.9141e-03  ...  -1.4364e-03 -2.2118e-03 -2.2050
e-03
  7.8954e-04  1.3881e-03  1.3692e-03  ...  -1.3803e-03 -2.1782e-03 -2.0131
e-03
      ⋮

( 4 , 0 ,.,.) =
 -1.9464e-04 -5.1891e-03 -4.7168e-03  ...  -1.5148e-03 -9.3508e-04 -8.6177
e-04
 -6.8735e-03 -9.0426e-03 -9.4968e-03  ...  -2.0001e-03 -2.0508e-03 -1.6906
e-03
 -2.3670e-03 -3.6803e-03 -8.9425e-03  ...  -2.3348e-03 -2.1254e-03 -1.7499
e-03
                ...                   ⋱                   ...

  3.3233e-03  6.1639e-03  6.7989e-03  ...  -4.8927e-03 -4.4033e-03 -4.0585
e-03
  2.9771e-03  5.6009e-03  5.5073e-03  ...  -4.4400e-03 -6.3114e-03 -6.1319
e-03
  2.4806e-03  4.1864e-03  4.0735e-03  ...  -4.0046e-03 -6.0211e-03 -5.5041
e-03
[torch.FloatTensor of size 5x1x640x640]
```

We can decide to sum score_accumulated_k or not!!

In [105]:

```
#vis.plot_scores(img_target,score_input.sum(1).unsqueeze(1), figsize=(50,50), only_positive
vis.plot_scores(0,score_input.sum(1).unsqueeze(0), figsize=(50,50), only_positive=False)
```



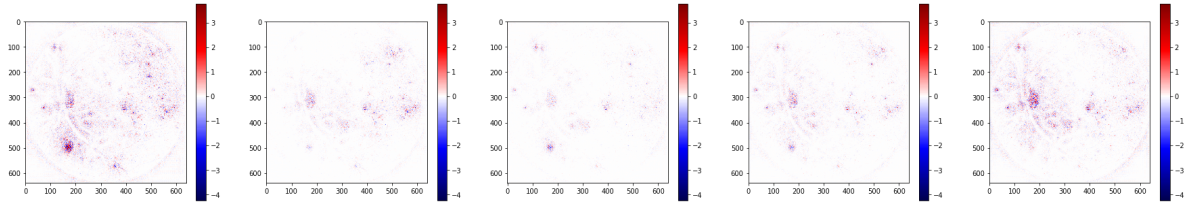
In [106]:

```
print("Part of the score represented by last layer:")
si = score_input.sum(3).sum(2).sum(1)
print(si)
sa = score_accumulated_k.sum(3).sum(2).sum(1).squeeze()
print(sa)
print(si+sa+me.lc[0].bias)
```

```
Part of the score represented by last layer:
Variable containing:
  5.9151
-46.6096
-15.8035
 16.4524
 49.9831
[torch.FloatTensor of size 5]

Variable containing:
-300.9490
  15.2049
  37.4547
 -50.9240
-244.5714
[torch.FloatTensor of size 5]

Variable containing:
-295.0830
 -30.9120
  21.7021
 -34.7909
-194.4052
[torch.FloatTensor of size 5]
```

In [107]:

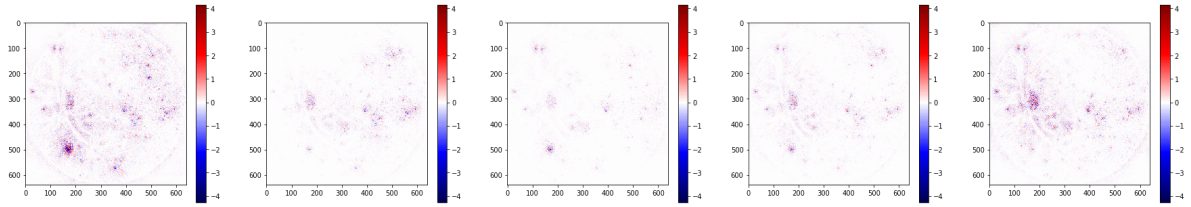
```
vis.plot_scores(0,score_input.sum(1).unsqueeze(0), figsize=(50,50))
print(score_input.sum(1).unsqueeze(0).sum(3).sum(2))
```



```
Variable containing:
  5.9151 -46.6096 -15.8035  16.4525  49.9831
[torch.FloatTensor of size 1x5]
```

In [108]:

```
score_total = score_input.sum(1).unsqueeze(0) + score_accumulated_k.squeeze().unsqueeze(0)
vis.plot_scores(0, score_total, figsize=(50,50))
print(score_total.sum(3).sum(2) + me.lc[0].bias)
print(score_total.size())
```



```
Variable containing:
-295.0829  -30.9120   21.7021  -34.7909 -194.4053
[torch.FloatTensor of size 1x5]

torch.Size([1, 5, 640, 640])
```

In [109]:

```
# Test for checking that the sum of the scores is equal to the last layer score (for every
nclasses = 5
for t in range(nclasses):
    total_score = score_k637[t].sum() + score_k509[t].sum() + score_k381[t].sum() + score_k
                  score_k189[t].sum() + score_k125[t].sum()+ score_k93[t].sum() + score_k61
                  score_k45[t].sum() + score_k29[t].sum() + score_k21[t].sum() + score_k13[
                  score_k9[t].sum() + score_k5[t].sum() + score_k3[t].sum() + score_input[t
    print("Class {} : {} == {} ".format(t, total_score.data[0], output[0][t].data[0]))

print("Score 637")
print(score_k637.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 509")
print(score_k509.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 381")
print(score_k381.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 253")
print(score_k253.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 189")
print(score_k189.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 125")
print(score_k125.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 93")
print(score_k93.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 61")
print(score_k61.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 45")
print(score_k45.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 29")
print(score_k29.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 21")
print(score_k21.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 13")
print(score_k13.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 9")
print(score_k9.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 5")
print(score_k5.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
print("Score 3")
print(score_k3.sum(3).sum(2).sum(1).data.unsqueeze(1).t())
```

```
Class 0 : -295.0830078125 == -295.0829772949219
Class 1 : -30.912036895751953 == -30.91199493408203
Class 2 : 21.70208168029785 == 21.702089309692383
Class 3 : -34.79093933105469 == -34.79090118408203
Class 4 : -194.4052734375 == -194.40524291992188
Score 637

 326.9360   84.2731   -9.3565  -51.2489 -158.9272
[torch.FloatTensor of size 1x5]

Score 509

-420.0283 -104.0317   16.5810   29.6840   45.5727
[torch.FloatTensor of size 1x5]

Score 381

 -27.1625  -56.5001  -59.5354   48.6623  200.5818
```

```
[torch.FloatTensor of size 1x5]

Score 253

 276.8928  150.9478    3.5843 -139.6199 -390.1153
[torch.FloatTensor of size 1x5]

Score 189

-139.2568  -29.5257  -14.4226  -33.6882   59.9352
[torch.FloatTensor of size 1x5]

Score 125

-227.8238   51.3346   92.2413   64.6814   56.4347
[torch.FloatTensor of size 1x5]

Score 93

 -46.5364  -87.5043  -39.6689   23.6418  141.9384
[torch.FloatTensor of size 1x5]

Score 61

-117.3893 -100.3311  -32.4600   -9.0328  -22.1459
[torch.FloatTensor of size 1x5]

Score 45

 97.9360  65.3845  40.1894   7.0174 -53.4240
[torch.FloatTensor of size 1x5]

Score 29

 48.4706 -22.8899 -15.0632 -34.1990 -30.5664
[torch.FloatTensor of size 1x5]

Score 21

-117.1392  -27.8789   30.5102   75.4954  106.3981
[torch.FloatTensor of size 1x5]

Score 13

 133.3633   36.8533   -4.8000  -19.0340  -86.5773
[torch.FloatTensor of size 1x5]

Score 9

-173.2654   48.7330   15.9142   -5.9334 -102.3684
[torch.FloatTensor of size 1x5]

Score 5

 85.1075 -52.8037   3.1432  17.4877  31.1993
[torch.FloatTensor of size 1x5]

Score 3

 -1.0534  59.1439  10.5974 -24.8378 -42.5070
[torch.FloatTensor of size 1x5]
```

In [110]:

```
std = torch.std(score_total.view(-1)).data[0]
pc = pred_class.data[0]
print("Score distribution for predicted class: {}".format(pc))
vis.plot_scores(pc,score_total.squeeze().unsqueeze(1), figsize=(100,100), only_positive=Tru
```

```
Score distribution for predicted class: 2
```