Report 5

Computational Neuroscience

Computer Assignment 5

Aref Afzali

610098014

In [1]:
```
from PIL import Image
import numpy as np
import torch
import matplotlib.pyplot as plt
```

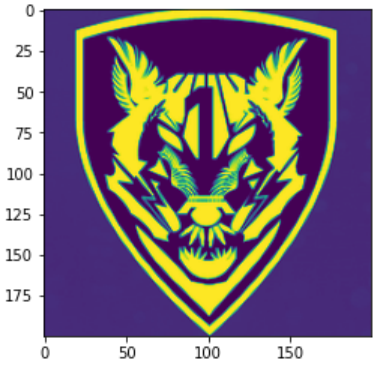
In [2]:
```
from cnsproject.encoding.encoders import Time2FirstSpikeEncoder, PoissonEncoder, Positio
nEncoder
from cnsproject.plotting.plotting import plotting
```

Walkthrough

In this part we first read the image. transform it into the grayscale form and reduce its resolution to 200*200 px and in the end transform it to a tensor.

In [3]:
```
im = Image.open("./img.jpg").convert('L')
im = im.resize((200,200),Image.ANTIALIAS)
data = torch.from_numpy(np.asarray(im))
plt.imshow(im)
plt.show()
```

```
<ipython-input-3-db8898f636c9>:3: UserWarning: The given NumPy array is not writeable, and PyTorch does not support non-writ
eable tensors. This means you can write to the underlying (supposedly non-writeable) NumPy array using the tensor. You may w
ant to copy the array to protect its data or make it writeable before converting it to a tensor. This type of warning will b
e suppressed for the rest of this program. (Triggered internally at  /pytorch/torch/csrc/utils/tensor_numpy.cpp:143.)
  data = torch.from_numpy(np.asarray(im))
```

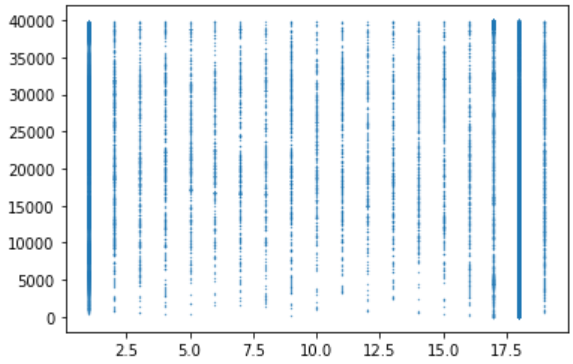


In this part we will make a encoder and encode the input tensor.

In [4]:
```
time = 20
encoder = Time2FirstSpikeEncoder(time = time)
I = encoder(data)
fi = torch.flatten(I, start_dim=1)
```

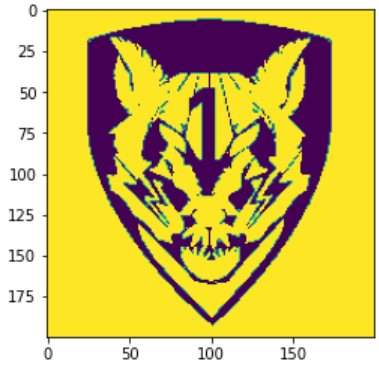
Then we will plot a roaster plot which shows the neuron's spikes in each unit of time.

In [5]:
```
sf = np.flipud(fi.T)
args = np.argwhere(sf)
plt.scatter(args.T[1,:], args.T[0,:], s=0.1)
plt.show()
```



In the end, we will decode the encoded matrix and then plot it to see the original picture.

In [6]:
```
x = encoder.decode(fi.numpy(), (200,200))
plt.imshow(x)
plt.show()
```



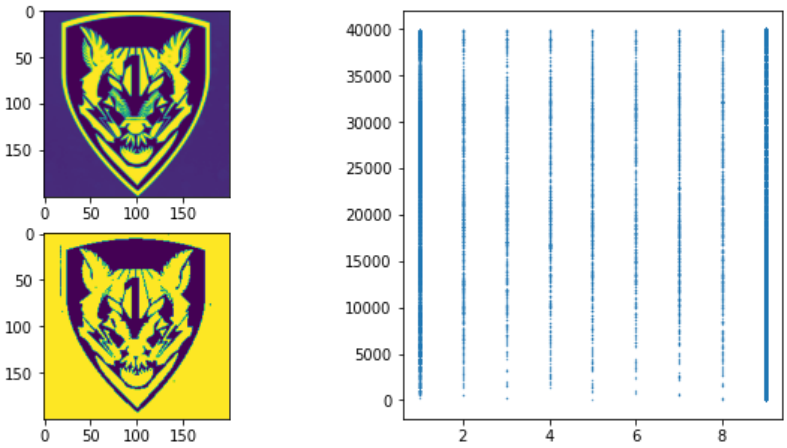
Time to First Spike Encoder

Like the previous part, in this part we are looking forward to check the Time2FirstSpikeEncoder.

The left images are before(up) the encoding and after(down) the decoding. The right plot is the encoded pattern.

In [7]:
```
time = 10
encoder = Time2FirstSpikeEncoder(time = time)
I = encoder(data)
fi = torch.flatten(I, start_dim=1)
x = encoder.decode(fi.numpy(), (200,200))
plot = plotting()
plot.plot_encoding_decoding(data,fi,x)
plot.show()
```

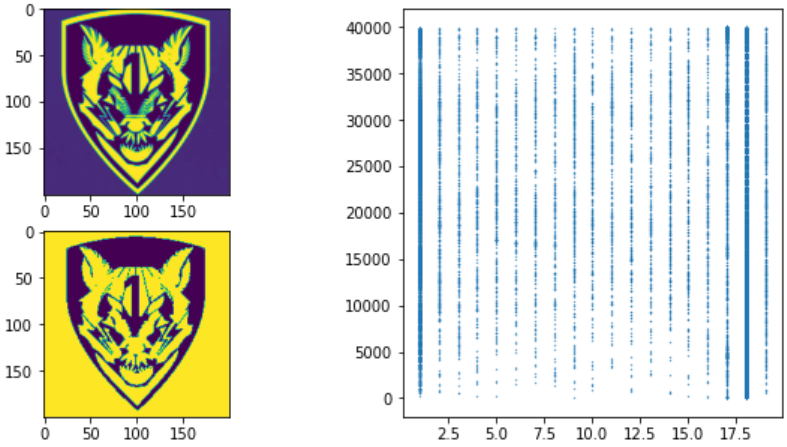
```
No handles with labels found to put in legend.
```



As we can see in the previous cell, it doesn't have all the details of the main image but on the other hand it doen't have any noise. For improving the output we can increase the time for encoding.

In [8]:
```
time = 20
encoder = Time2FirstSpikeEncoder(time = time)
I = encoder(data)
fi = torch.flatten(I, start_dim=1)
x = encoder.decode(fi.numpy(), (200,200))
plot = plotting()
plot.plot_encoding_decoding(data,fi,x)
plot.show()
```

```
No handles with labels found to put in legend.
```

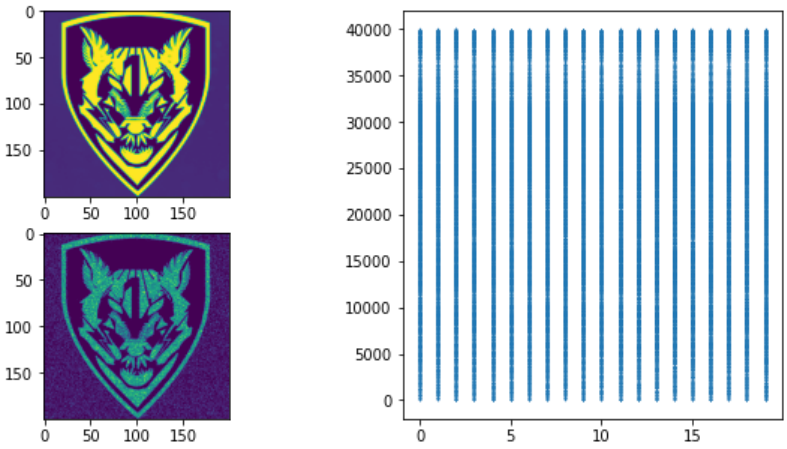


Poisson Encoder

In this part we are looking forward to check the PoissonEncoder.

In [9]:
```
time = 20
encoder = PoissonEncoder(time = time, r = 10)
I = encoder(data)
fi = torch.flatten(I, start_dim=1)
x = encoder.decode(fi.numpy(), (200,200))
plot = plotting()
plot.plot_encoding_decoding(data,fi,x)
plot.show()
```

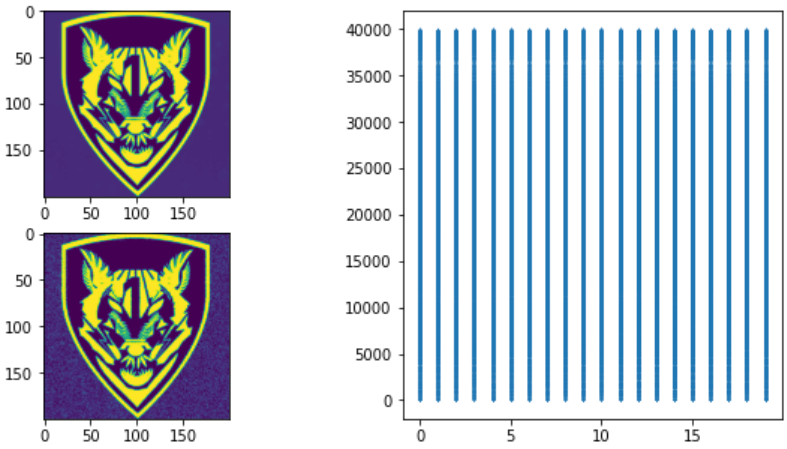
```
No handles with labels found to put in legend.
```



As we can see in the previous cell, there is some noise in the decoded image but on the other hand it shows the details better. For improving the output we can increase the r(scale rate) for encoding.

In [10]:
```
time = 20
encoder = PoissonEncoder(time = time, r = 20)
I = encoder(data)
fi = torch.flatten(I, start_dim=1)
x = encoder.decode(fi.numpy(), (200,200))
plot = plotting()
plot.plot_encoding_decoding(data,fi,x)
plot.show()
```

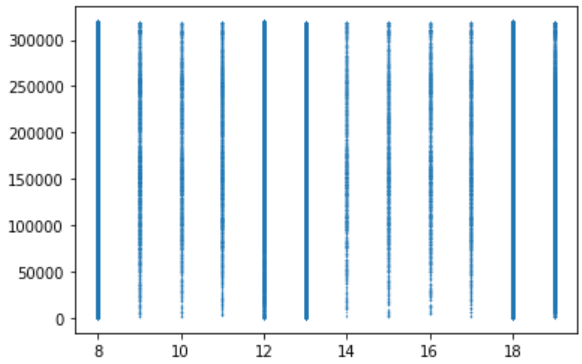
```
No handles with labels found to put in legend.
```



Representing Number

In this part we are looking forward to check the PositionEncoder.

In [11]:
```
time = 20
neuron_numbers = 8
neuron_range_mean = np.arange(1,neuron_numbers+1)
neuron_range_std = [1]*neuron_numbers
encoder = PositionEncoder(time = time, neuron_numbers=neuron_numbers,
    neuron_range_mean=neuron_range_mean, neuron_range_std=neuron_range_std)
I = encoder(data)
fi = torch.flatten(I, start_dim=1)
sf = np.flipud(fi.T)
args = np.argwhere(sf)
plt.scatter(args.T[1,:], args.T[0,:], s=0.1)
plt.show()
```

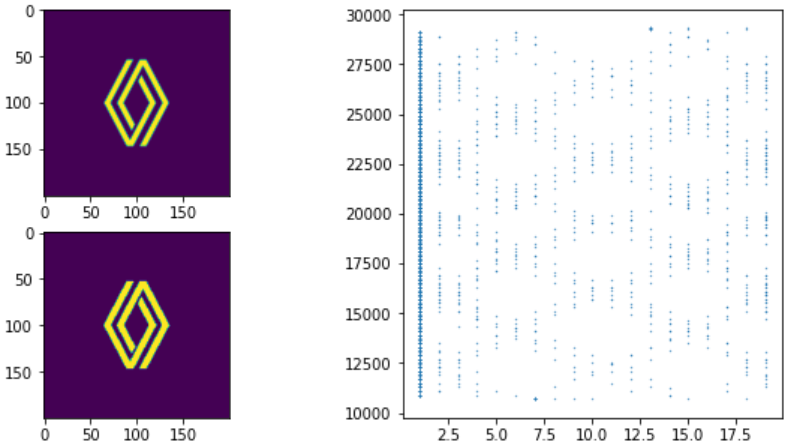


For another image

Time2First

In [12]:
```
im = Image.open("./img2.jpg").convert('L')
im = im.resize((200,200),Image.ANTIALIAS)
data = torch.from_numpy(np.asarray(im))
time = 20
encoder = Time2FirstSpikeEncoder(time = time)
I = encoder(data)
fi = torch.flatten(I, start_dim=1)
x = encoder.decode(fi.numpy(), (200,200))
plot = plotting()
plot.plot_encoding_decoding(data,fi,x)
plot.show()
```

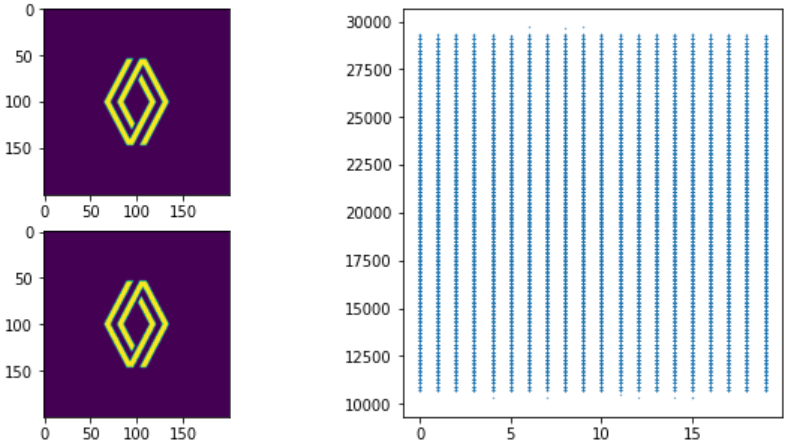
```
No handles with labels found to put in legend.
```



Poisson

In [13]:
```
time = 20
encoder = PoissonEncoder(time = time, r=20)
I = encoder(data)
fi = torch.flatten(I, start_dim=1)
x = encoder.decode(fi.numpy(), (200,200))
plot = plotting()
plot.plot_encoding_decoding(data,fi,x)
plot.show()
```

```
No handles with labels found to put in legend.
```



Numbers

In [14]:
```
time = 20
neuron_numbers = 8
neuron_range_mean = np.arange(1,neuron_numbers+1)
neuron_range_std = [1]*neuron_numbers
encoder = PositionEncoder(time = time, neuron_numbers=neuron_numbers,
    neuron_range_mean=neuron_range_mean, neuron_range_std=neuron_range_std)
I = encoder(data)
fi = torch.flatten(I, start_dim=1)
sf = np.flipud(fi.T)
args = np.argwhere(sf)
plt.scatter(args.T[1,:], args.T[0,:], s=0.1)
plt.show()
```