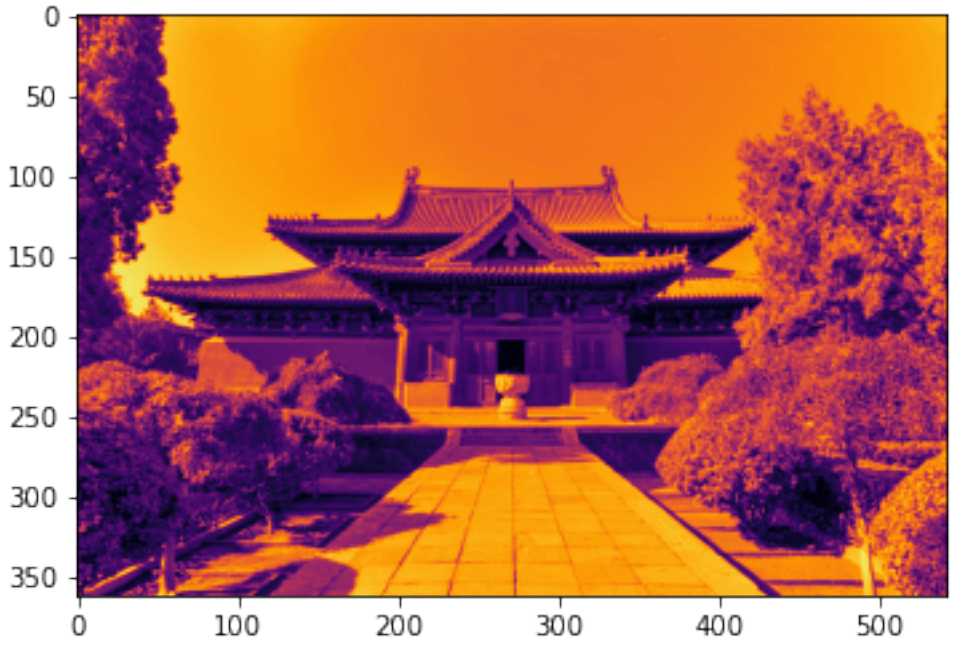
1 Boxcar intensity-level entropy

```
import numpy as np
import numpy.linalg as linalg
import matplotlib.pyplot as plt
from PIL import Image, ImageFilter, ImageOps
from src.utilities import *
from src.intensity_entropy import *
from src.kernels import *
plt.rcParams['image.cmap'] = 'inferno'
```

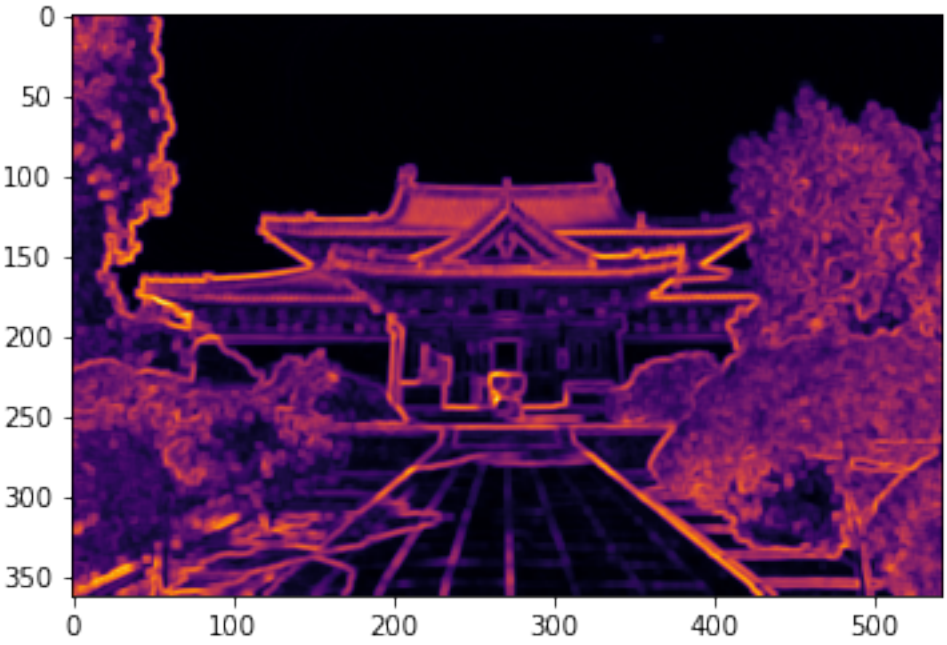
Let's compare the boxcar images for intensity entropy to those for a positive function on an image (the standard deviation) and for different functions of the induced intensity distribution.

```
img = ImageOps.grayscale(Image.open('test.jpg'))
scale = max(np.shape(img))
data = np.array(img)
plt.imshow(img);
```



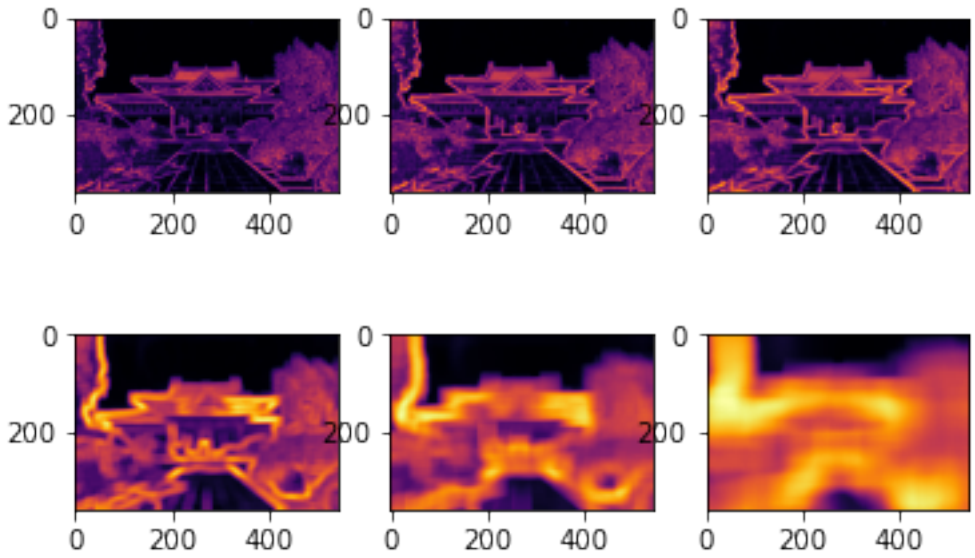
1.1 Standard deviation

```
plt.imshow(mapbox(2, np.std, np.array(img)));
```



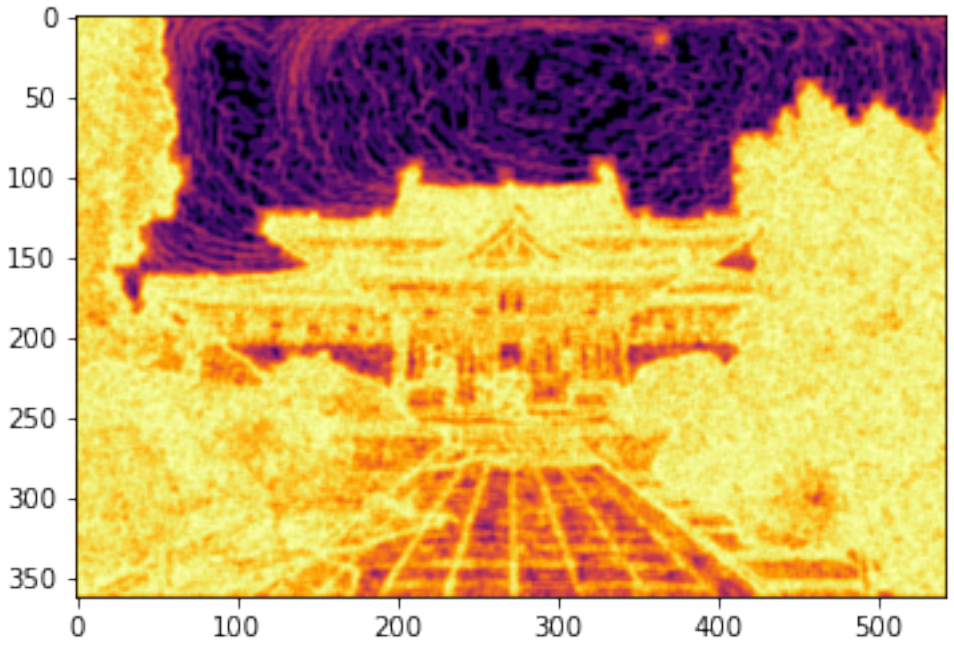

```
boxos = list(mapboxes([1,2,3,10,20,50], np.std, np.array(img)))
```

```
_, axarr = plt.subplots(2, np.ceil(len(boxos)/2).astype('int'))
for i, subimg in enumerate(boxos[:3]):
    axarr[0,i].imshow(subimg)
for i, subimg in enumerate(boxos[3:]):
    axarr[1,i].imshow(subimg)
plt.show()
```



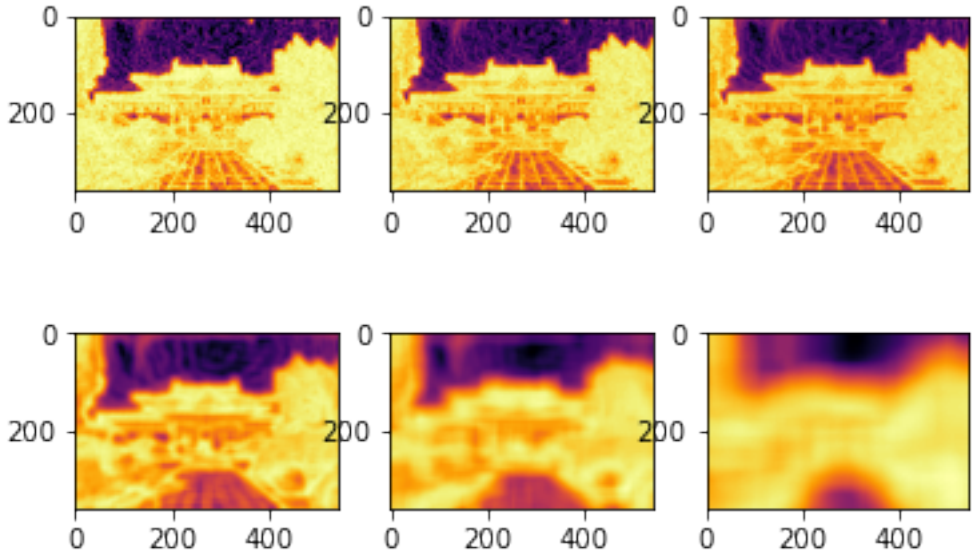
1.2 Intensity entropy

```
plt.imshow(mapbox(2, intensity_entropy, np.array(img)));
```



```
boxSes = list(mapboxes([1,2,3,10,20,50], intensity_entropy, np.array(img)))
```

```
_, axarr = plt.subplots(2, np.ceil(len(boxSes)/2).astype('int'))
for i, subimg in enumerate(boxSes[:3]):
    axarr[0,i].imshow(subimg)
for i, subimg in enumerate(boxSes[3:]):
    axarr[1,i].imshow(subimg)
plt.show()
```

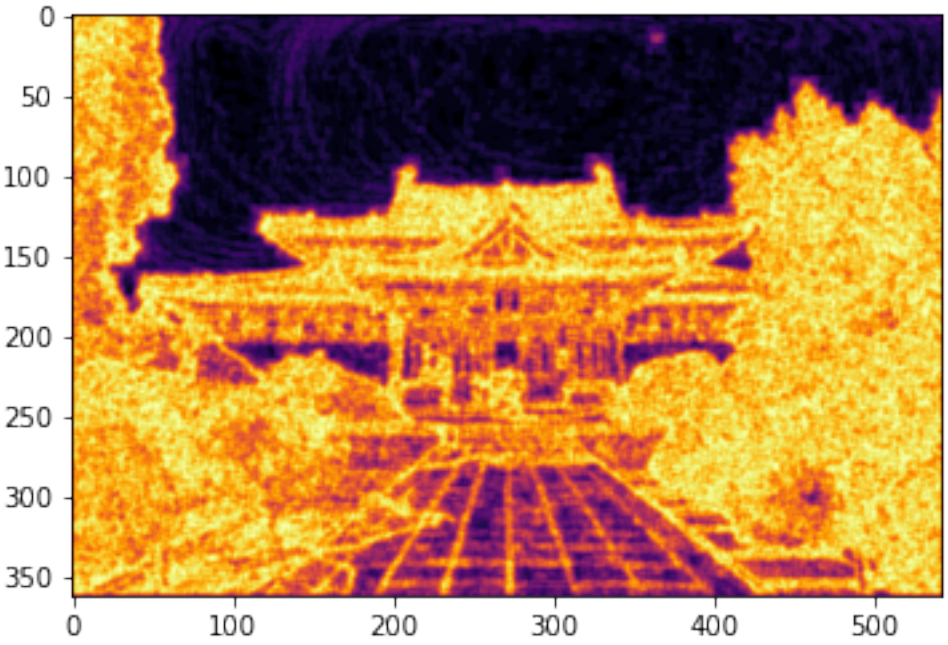


1.3 Replace surprisal with other functions

To what extent do the surprisal-related results depend upon the specific form of the *surprisal* $x \mapsto -\log(x)$ in the expected value of the intensity distribution? We will replace the expected surprisal with the expected f, for different functions f on the empirical probabilities of a pixel taking some intensity.

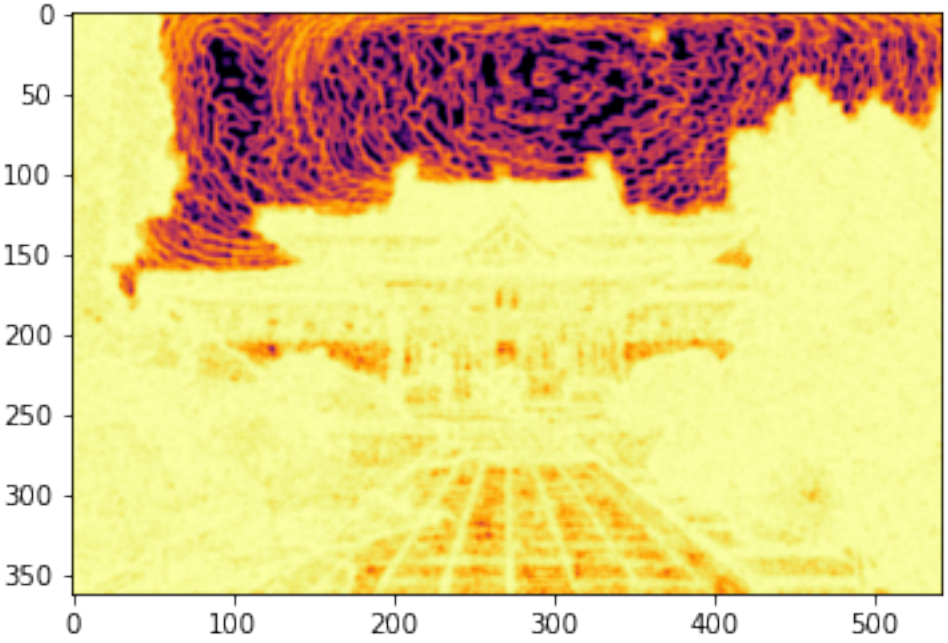
Laurent: $p \mapsto -1 + 1/p$.

```
plt.imshow(mapbox(2, lambda I: intensity_expected(lambda p: -1 + 1/p if p > 0 else 0, I),
↪  np.array(img)));
```



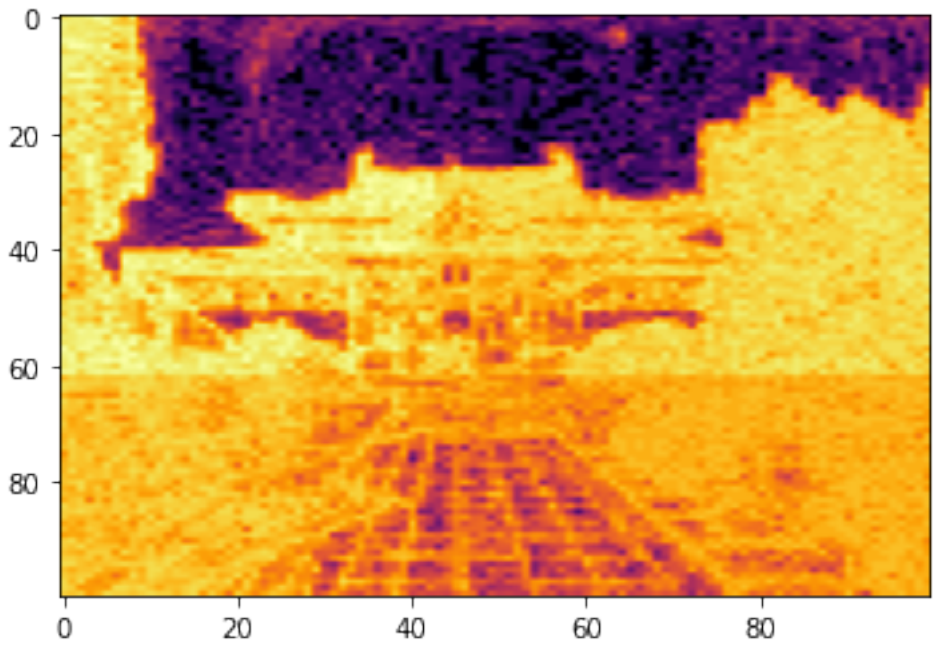
Taylor: $p \mapsto -(1+p)$.

```
plt.imshow(mapbox(2, lambda I: intensity_expected(lambda p: -(1+p), I), np.array(img)));
```

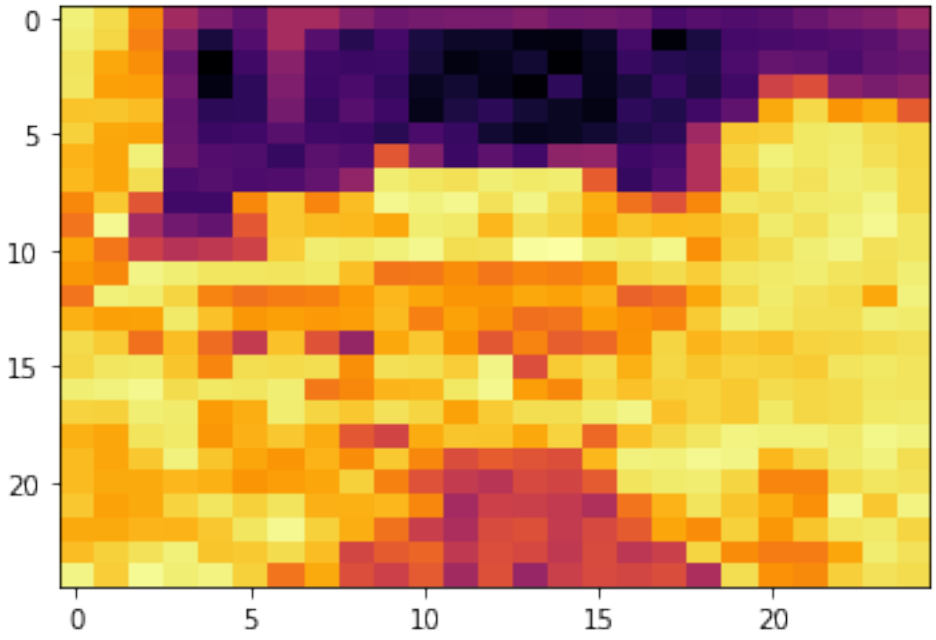


1.4 Intensity entropy on disjoint blocks

```
plt.imshow(mapblocks(100, 100, intensity_entropy, np.array(img)),
           aspect=np.divide(*np.shape(img)));
```




```
plt.imshow(mapblocks(25, 25, intensity_entropy, np.array(img)),
           aspect=np.divide(*np.shape(img)));
```


0
5
10
15
20
0
5
10
15
20