1 Effect of smoothing on intensity-level entropy

```python
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 *
```

1.1 Natural image

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

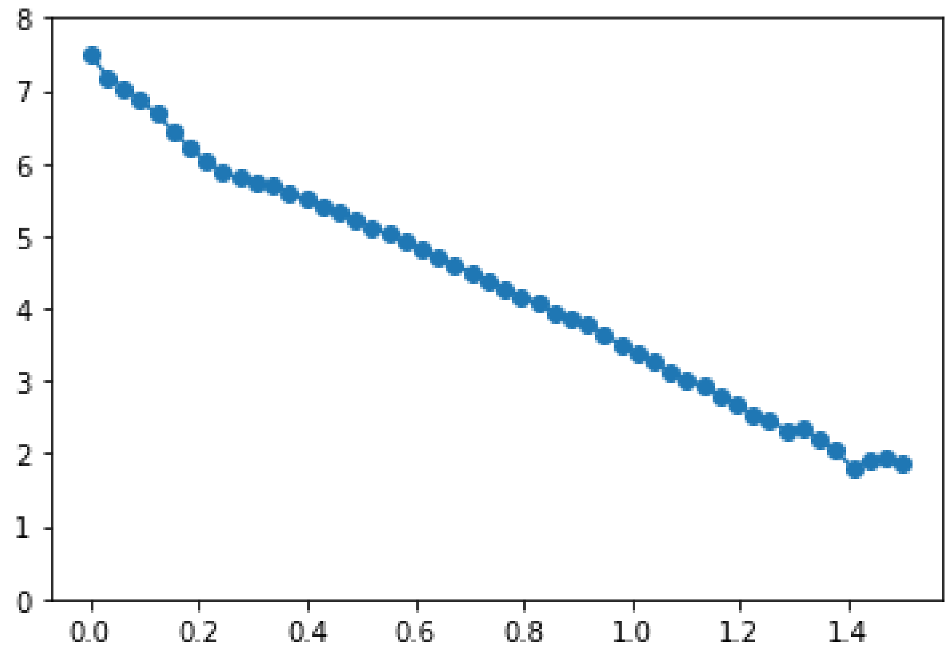
```python
intensity_entropy(img)
```

```
7.51132356216608
```

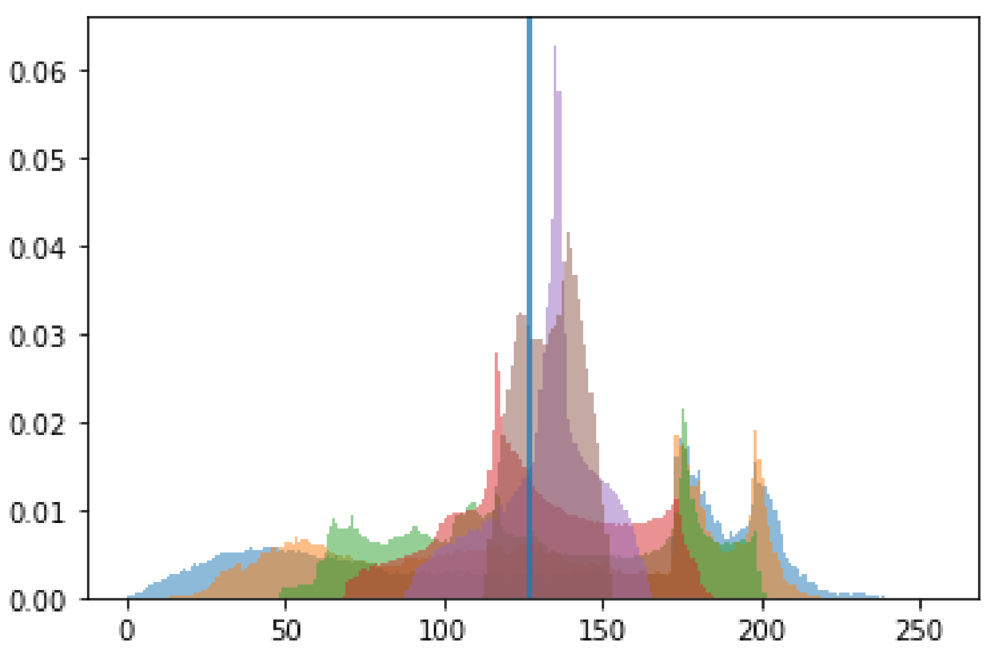
The problem with the intensity entropy is that it is usually near maximum (8 bits for these grayscale images).

```python
def intensity_blur(img, scales, display=True):
    scale = max(np.shape(img))

    results = []
    for k in scales:
        simg = img.filter(ImageFilter.GaussianBlur(k * scale))
        data = np.array(simg)
        ihist, ibins = np.histogram(data, bins=range(256+1), density=True)
        S = shannon_entropy(ihist)
        if display:
            hist = plt.hist(ibins[:-1], ibins, weights=ihist, alpha=0.5)
            results.append((k, simg, hist, S))
        else:
            results.append((k, S))

    if display:
        plt.axvline(x=np.mean(np.array(img)))

    return results
```

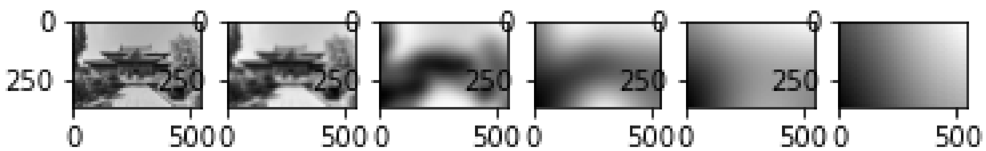
```python
results = intensity_blur(img, np.linspace(0, 1.5, num=50), False)

plt.plot(*np.transpose(results), 'o-')
plt.ylim((0, 8))
plt.xlabel = "Smoothing"
plt.ylabel = "Intensity Entropy (bits)"
```




```
rimgs = [img for _, img, _, _ in intensity_blur(img, [0, 0.01, 0.05, 0.125, 0.25, 0.5])]
plt.show()
```

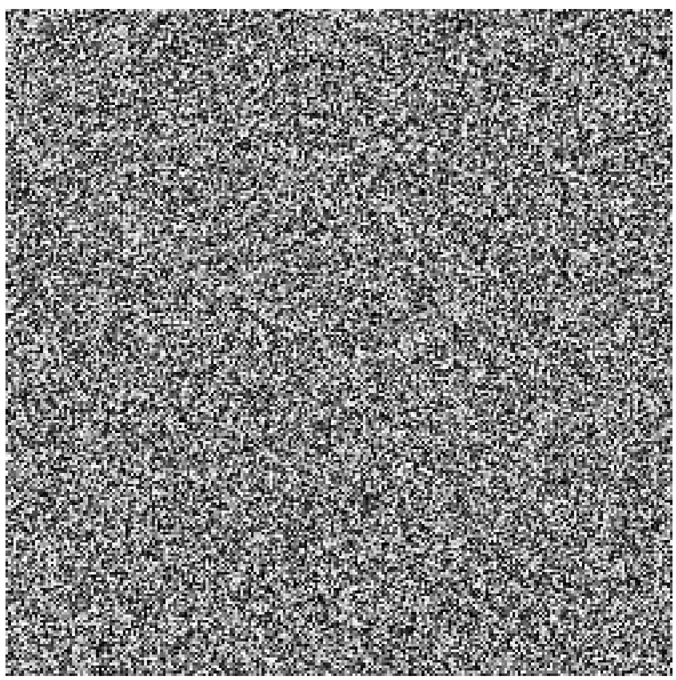


```
_, axarr = plt.subplots(1, len(rimgs))
for i, subimg in enumerate(rimgs):
    axarr[i].imshow(subimg, cmap='gray')
plt.show()
```



1.2 Random pixel values

```
rsize = 250
randimg = Image.fromarray((256*np.random.rand(*2*[rsize])).astype('uint8'))
randimg
```

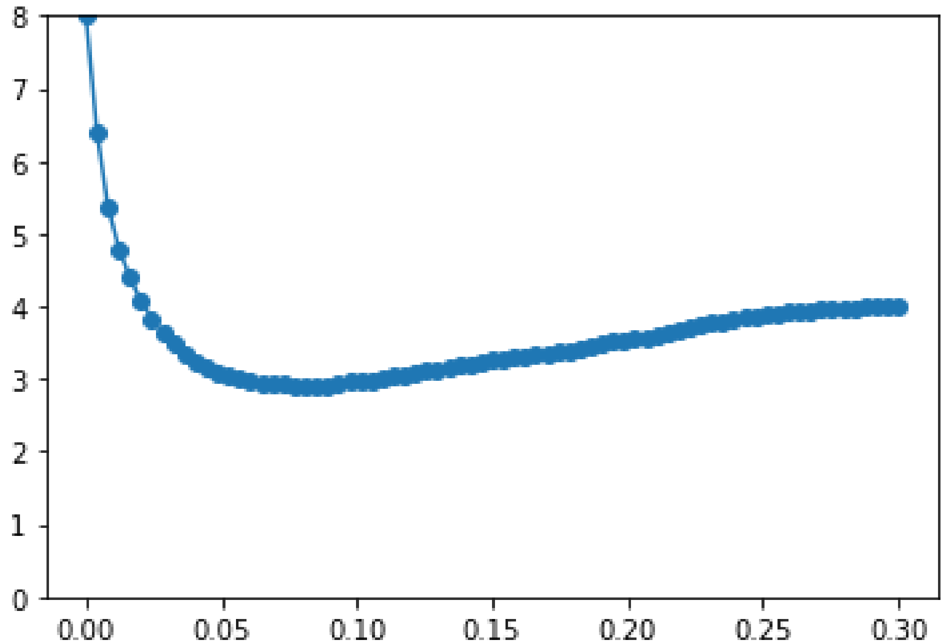


1.2.1 Beware: GIGO

The boundary effects and discrete kernel of `ImageFilter.GaussianBlur` renders the data unreliable after the "minimum" of the intensity entropy with smoothing.

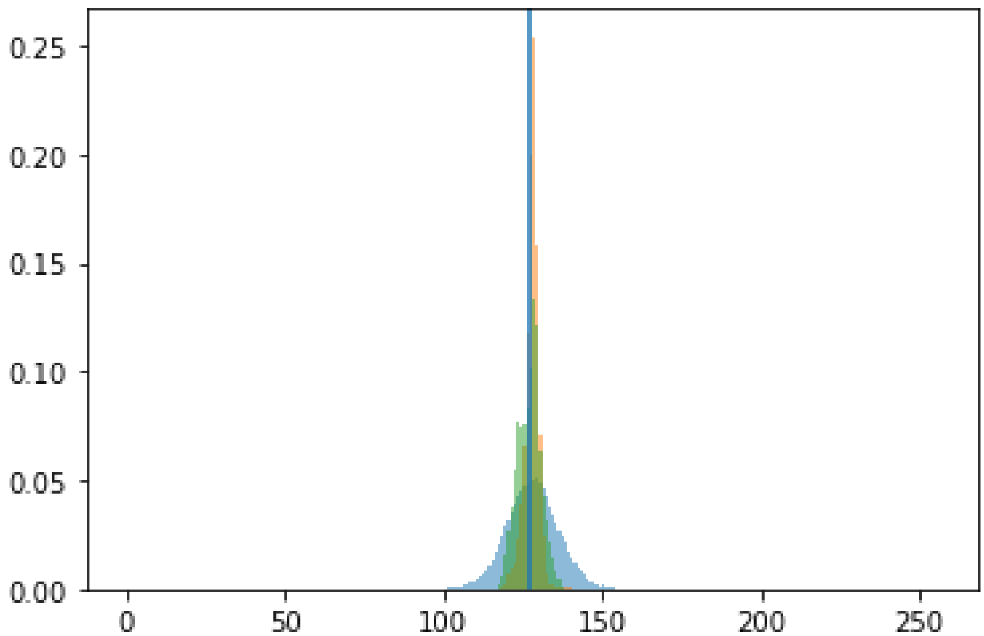
This is immediately clear after even small smoothing for random pixel values, since there are no spatial correlations.

```
results = intensity_blur(randimg, np.linspace(0, 0.3, num=75), False)

plt.plot(*np.transpose(results), 'o-')
plt.ylim((0, 8))
plt.xlabel = "Smoothing"
plt.ylabel = "Intensity Entropy (bits)"
```

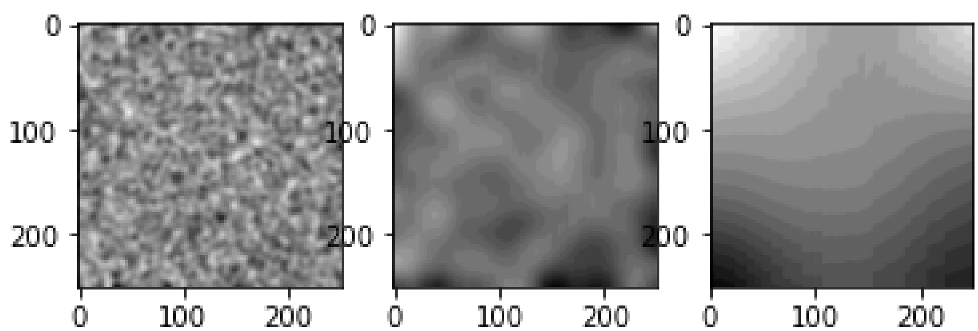



```
rimgs = [img for _, img, _, _ in intensity_blur(randimg, [0.01, 0.05, 0.25])]
```

```
plt.show()
```




```python
_, axarr = plt.subplots(1, len(rimgs))
for i, subimg in enumerate(rimgs):
    axarr[i].imshow(subimg, cmap='gray')
plt.show()
```



The rightmost image should be uniform: the renormalization emphasizes incorrect deviations. These are what keep the intensity entropy from vanishing.

1.3 Comparing different levels of smoothing

Is composing n Gaussian blurs with variance σ^2 the same as doing one with variance $n\sigma^2$ (considering the boundary effects and discrete kernel)?

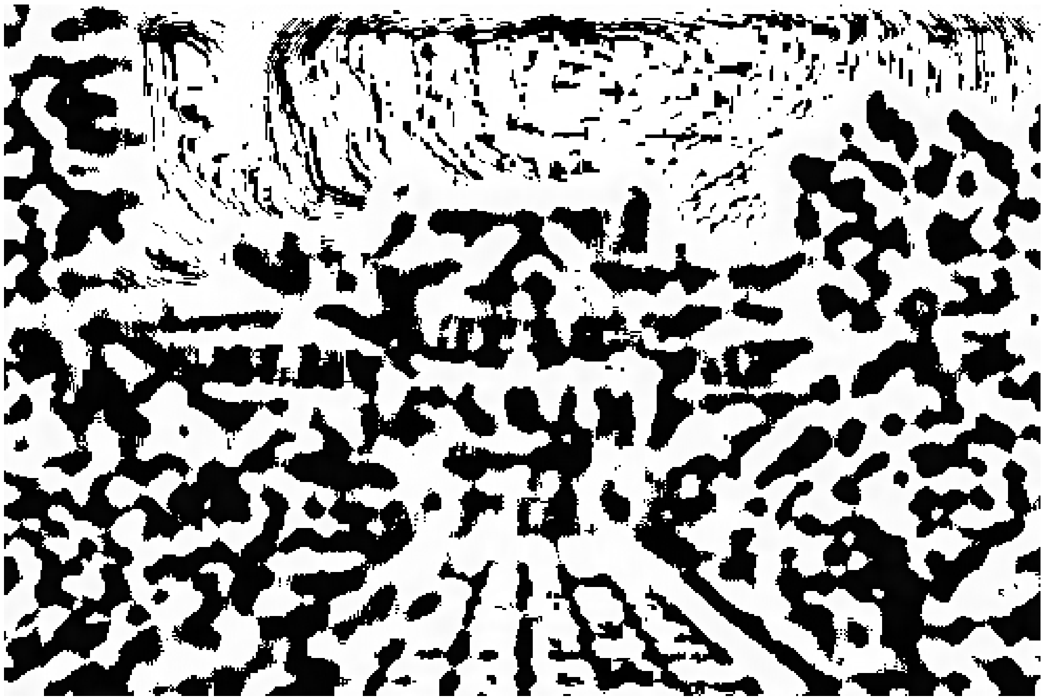
```python
nsmooths = 10
cimg = img
oneimg = cimg.filter(ImageFilter.GaussianBlur(np.sqrt(nsmooths)*2))
oneimg
```

```python
nimg = cimg
for _ in range(nsmooths):
    nimg = nimg.filter(ImageFilter.GaussianBlur(2))
nimg
```

Answer: **No**

The differences between results at different scales can be pretty wack.

```
Image.fromarray((255*rescale(np.array(nimg) - np.array(oneimg))).astype('uint8'))
```



```
smimg = img
smdiff = np.array(smimg.filter(ImageFilter.GaussianBlur(2))) -
↪ np.array(smimg.filter(ImageFilter.GaussianBlur(100)))
diffimg = Image.fromarray((255 * rescale(smdiff)).astype('uint8'))
diffimg
```