6.689 Problem Set 7 – Bennett Hellman

Problem 1)

i) Compute the Gaussian:

```
G = np.zeros((w, w)) # Fill in with your code
    def Gauss_kernel_filter(kern_dim, mu, sigma):
      x, y = np.meshgrid(np.linspace(-kern_dim/2, kern_dim/2,
kern_dim), np.linspace(-kern_dim/2, kern_dim/2, kern_dim))
      Gauss_filt = np.exp(-1*((x-mu)**2+(y-mu)**2)/(2*sigma**2))
      return Gauss_filt
    G = Gauss_kernel_filter(w, 0, w/6.4)
    G = np.repeat(G[:, :, np.newaxis], nChannels, axis=2)
```

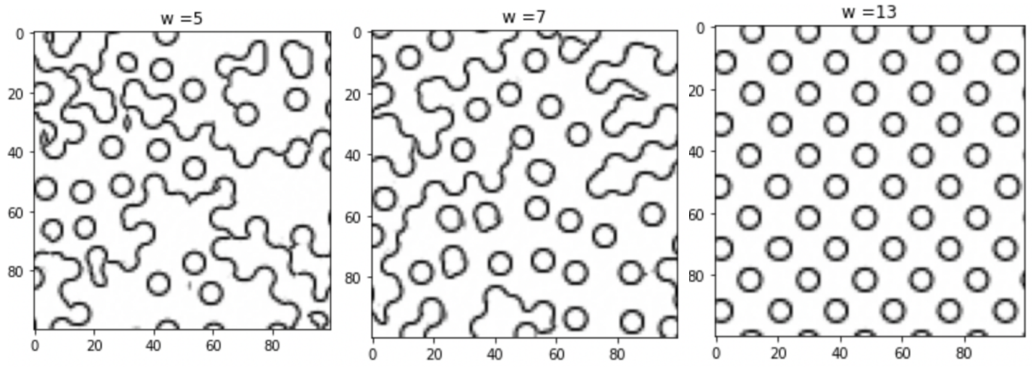
ii) Find matches function:

```
def Find_matches(template, sample, G):
    validMask = np.where(~np.isnan(template), 1, 0)
    mask=G*validMask
    mask = (mask/(np.sum(mask, axis = (0,1)))).ravel(order='F')
    r_mat = im2col_sliding(sample[:,:,0], template[:,:,0].shape)
    g_mat = im2col_sliding(sample[:,:,1], template[:,:,1].shape)
    b_mat = im2col_sliding(sample[:,:,2], template[:,:,2].shape)
    rgb_mat = np.stack((r_mat, g_mat, b_mat), axis=2)
    rgb_mat = np.vstack([rgb_mat[:,i,:].ravel(order='F') for i in
range(rgb_mat.shape[1])])
    temp_rav = template.ravel(order='F')
    SSD = np.nansum((mask*(rgb_mat-temp_rav)**2), axis=1)
    min_error = SSD.min()
    g_idx =
np.argwhere((SSD<=delta)&(SSD<=min_error*(1+epsilon)))[:,0]
    g_cols = rgb_mat[g_idx]
    best_matches = np.stack([np.reshape(g_cols[i,:],
                        template.shape, order =
'F')[np.shape(template)[0]//2, np.shape(template)[0]//2] for i in
range(g_cols.shape[0])]).T
    errors = SSD[g_idx] # Fill in with your code
    return best_matches, errors
```

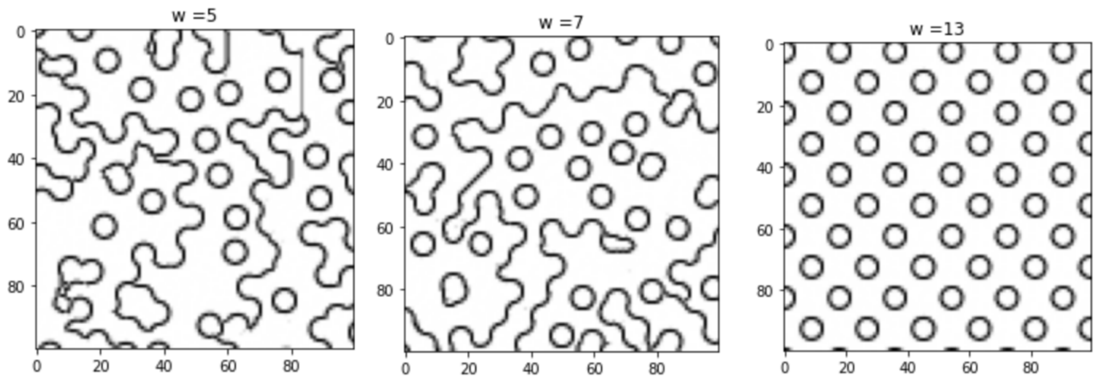
iii) Sample from the output:

```
synthIm[ii[i], jj[i],:] =
best_matches[:,np.random.choice(range(best_matches.shape[1]))]
```

Output:



As the window size increases, the algorithm's performance increases. This makes intuitive sense because a larger window gives the algorithm more information to infer on, specifically more information about the global structure of the texture. Since we are drawing randomly from our best matches, new initializations will cause different outputs based on the window size, although they will be similar looking. The smaller the window, the more a reinitialization's output can deviate. Once the window size is large enough, the algorithm understands enough of the global structure that it can only fill it in the same way with small deviations. As you can see from the below reinitialized pictures, w=5 and w=7 are much different form their predecessors but w=7 performs slightly better in both. However, w=13 is only slightly different, and always the best performing, providing evidence for my point.



Problem 2)

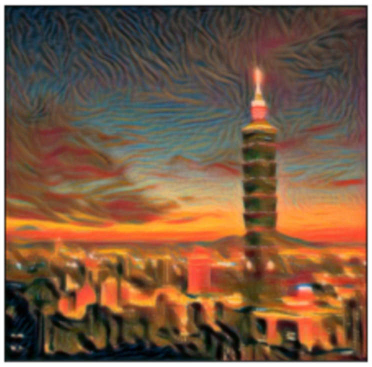
1. Gram matrix:

```
def gram_matrix(input):
    a, b, c, d = input.size()
    features = input.view(a*b,c*d)
    G =  torch.matmul(features, features.t())
    return G.div(a * b * c * d)
```

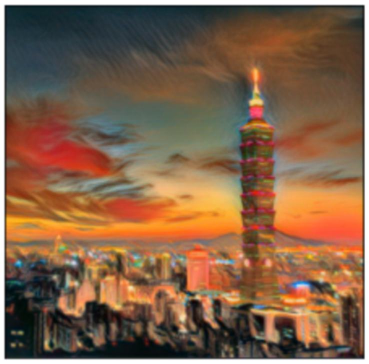
2. The answer is both A & B. Style-loss shouldn’t encode the spatial correspondence between the optimized image and the style-image because we do want the actual structure of the image. By doing feature correlation over space, we do not get any global structures. Style-loss also shouldn’t match the distribution of features found in the optimized image and the style-image because that is what the content-loss does.

3. The answer is both B & C. Convolution 1 features care about microscopic structure because shallower layers look at more granular structures (e.g., edges & lines) while latter layers look at larger structure (e.g., large shapes and patterns).

Conv1:



Conv5:



4. Johnson et al. are focused on perceptual loss, as opposed to pixel per-pixel loss, based on high-level features. They utilize both an image transform net and a loss network. The former converts an input image into an output image. Conversely, the loss network is one pretrained and static during the training process, but it defines perceptual loss functions. They can leverage pretrained networks because they have learned to encode semantic information in an image. Perceptual loss functions measure the differences between content and style images. The paper demonstrates similar qualitative results, but their method is complete by the time it takes a single optimization step to complete in the baseline method.