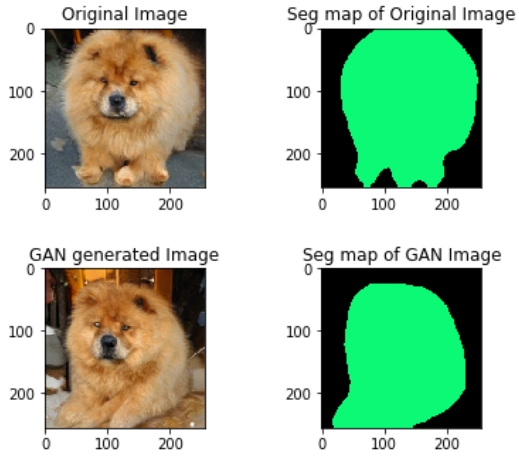
In [9]:

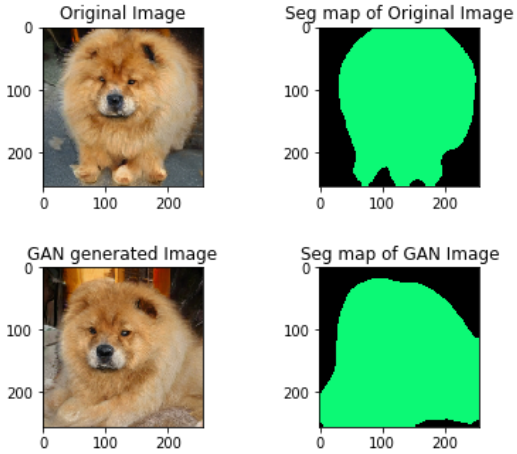
```
biggan.eval()
# optimizer = torch.optim.AdamW([z], lr=0.1)
optimizer = optim.Adam([z], lr=1e-1, betas=(0.5, 0.999))

# create a color pallette, selecting a color for each class
palette = torch.tensor([2 ** 25 - 1, 2 ** 15 - 1, 2 ** 21 - 1])
colors = torch.as_tensor([i for i in range(21)])[:, None] * palette
colors = (colors % 255).numpy().astype("uint8")

for i in range(20):
    with torch.enable_grad():
        optimizer.zero_grad()
        img = biggan(z, c, truncation.item())
        output_predicted_image = model(0.5 * (img.cuda() + 1))['out'][0]
        output_original_image = model(input_batch)['out'][0]
    #     output_predicted_image = output.argmax(0).float()
        loss = torch.nn.CrossEntropyLoss()
    #         cost = torch.sqrt(loss(output_predicted_image, output_original_image))
        cost = loss(output_predicted_image.unsqueeze(0), output_original_image.argmax(0
).unsqueeze(0))
        print("cost",i+1," is ", cost)

    fig = plt.figure()
    plt.subplots_adjust(top = 0.99, bottom=0.01, hspace=0.5, wspace=0.4)

    ax1 = fig.add_subplot(221)
    ax1.title.set_text('Original Image')
    plt.imshow(input_image)

    ax2 = fig.add_subplot(222)
    ax2.title.set_text('Seg map of Original Image')
    r = Image.fromarray(model(input_batch)['out'][0].argmax(0).float().byte().cpu().num
py())#.resize(input_image.size)
    r.putpalette(colors)
    plt.imshow(r)

    ax3 = fig.add_subplot(223)
    ax3.title.set_text('GAN generated Image')
    pil = torchvision.transforms.ToPILImage()((0.5 * (img.data + 1)).squeeze())
    plt.imshow(pil)

    ax4 = fig.add_subplot(224)
    ax4.title.set_text('Seg map of GAN Image')
    r = Image.fromarray(output_predicted_image.argmax(0).float().byte().cpu().numpy())
    r.putpalette(colors)
    plt.imshow(r)

    plt.show()

    print('-'*100)
    cost.backward()
    optimizer.step()

print('End')
```



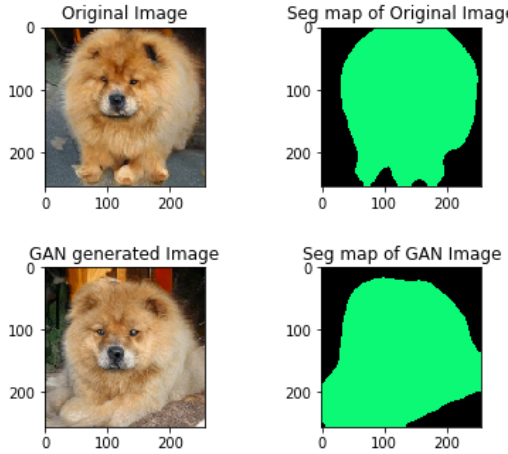
```
cost 1  is  tensor(0.8033, device='cuda:0', grad_fn=<NllLoss2DBackward>)
```



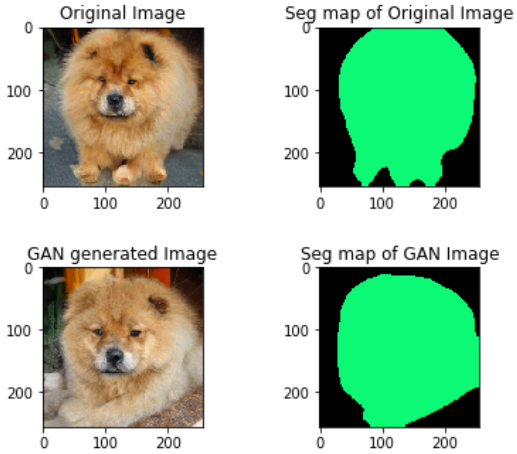

```
--------------------------------------------------------------------------------
--------------------------
cost 2  is  tensor(0.8158, device='cuda:0', grad_fn=<NllLoss2DBackward>)
```



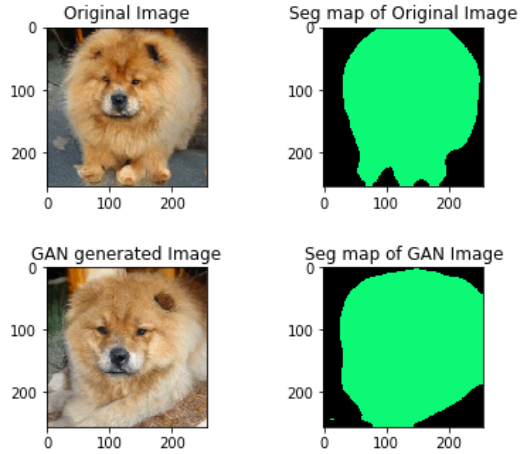
---
cost 3 is tensor(0.6636, device='cuda:0', grad_fn=<NllLoss2DBackward>)



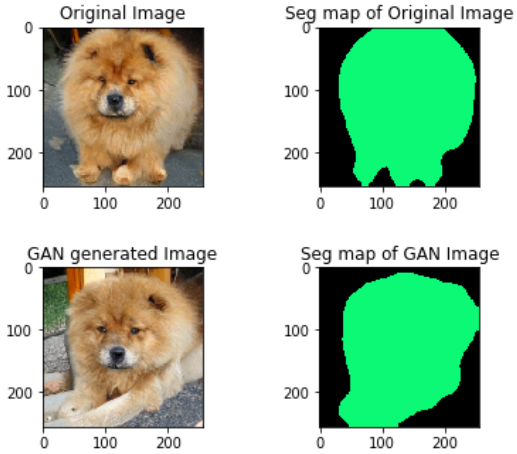
---
cost 4 is tensor(0.3577, device='cuda:0', grad_fn=<NllLoss2DBackward>)



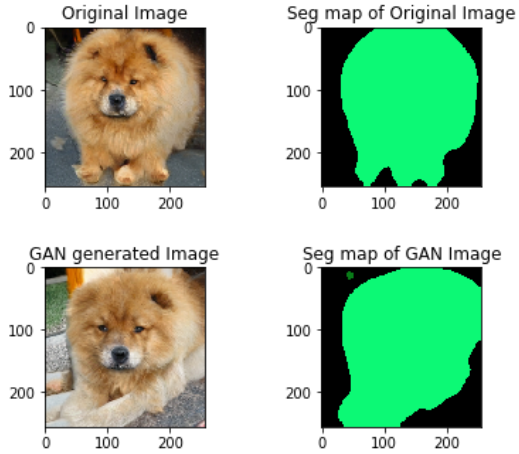
---
cost 5 is tensor(0.2919, device='cuda:0', grad_fn=<NllLoss2DBackward>)



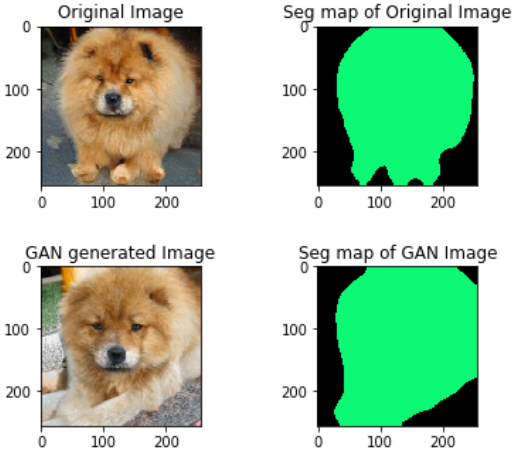
---
cost 6 is tensor(0.3218, device='cuda:0', grad_fn=<NllLoss2DBackward>)



---
cost 7 is tensor(0.2735, device='cuda:0', grad_fn=<NllLoss2DBackward>)



---
cost 8 is tensor(0.2814, device='cuda:0', grad_fn=<NllLoss2DBackward>)




```
--------------------------------------------------------------------------
-------------------------
cost 9  is  tensor(0.2292, device='cuda:0', grad_fn=<NllLoss2DBackward>)
```