In [10]:
```
import sys
```

In [11]:
```
import torch
import torch.nn.functional as F
from unet import UNet
```

In [12]:
```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = UNet(n_classes=2, padding=True, up_mode='upconv').to(device)
optim = torch.optim.Adam(model.parameters())
epochs = 10
```

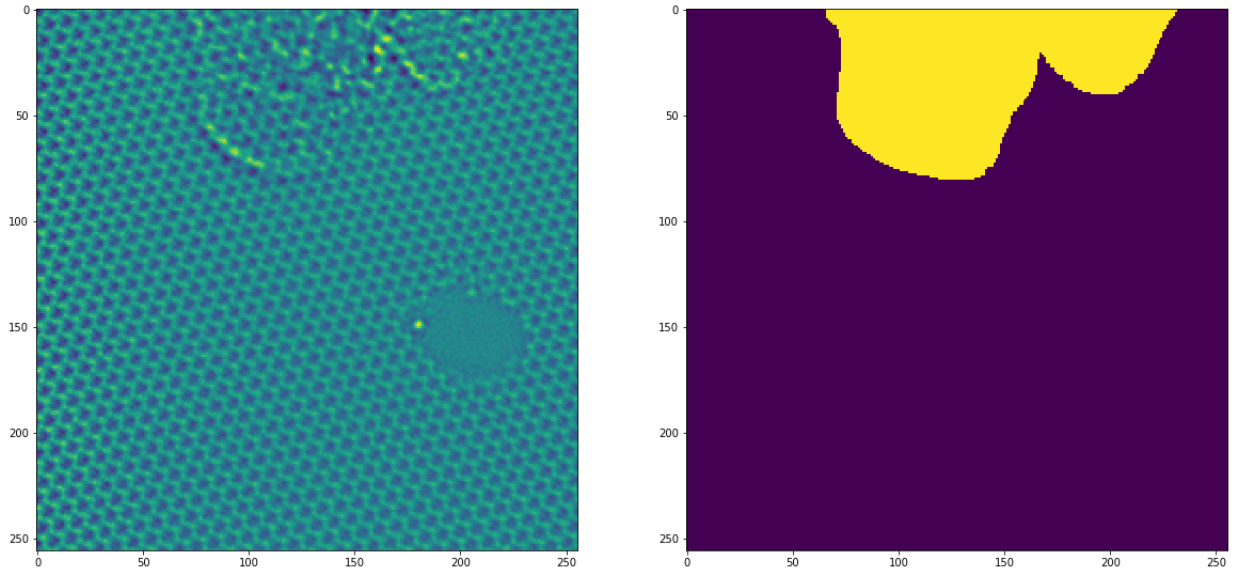
In [ ]:
```

```

In [ ]:
```

```

In [23]:
```
import matplotlib.pyplot as plt
import numpy as np
D = plt.imread("DATA.png")
L = plt.imread("LABEL.png")

print(D.shape)
L = L[:,:,0]

D = D/np.max(D)
L = (L/np.max(L)).astype(int)
```

```
(256, 256)
```

In [24]:
```
plt.figure(figsize = (20,20))
plt.subplot(1,2,1)
plt.imshow(D)
plt.subplot(1,2,2)
plt.imshow(L)
plt.show()
print('pass')
```

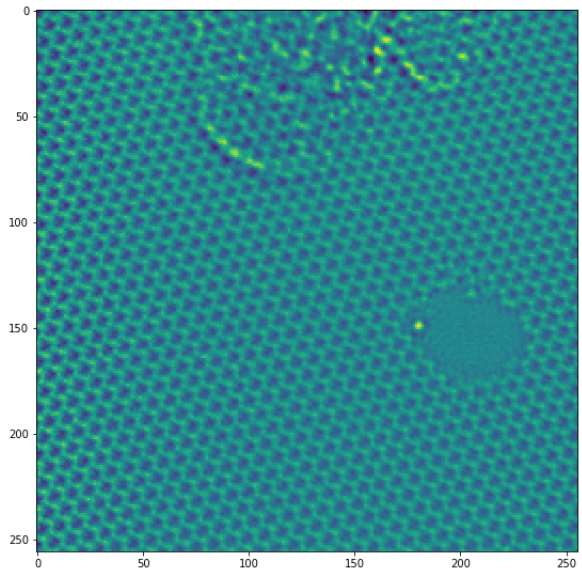
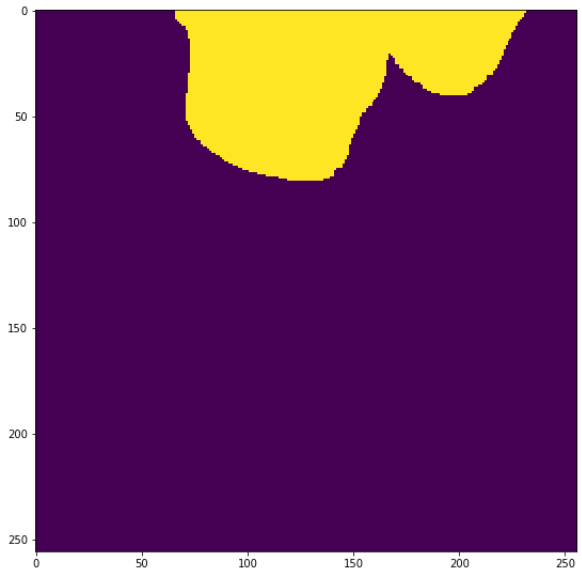


```
pass
```

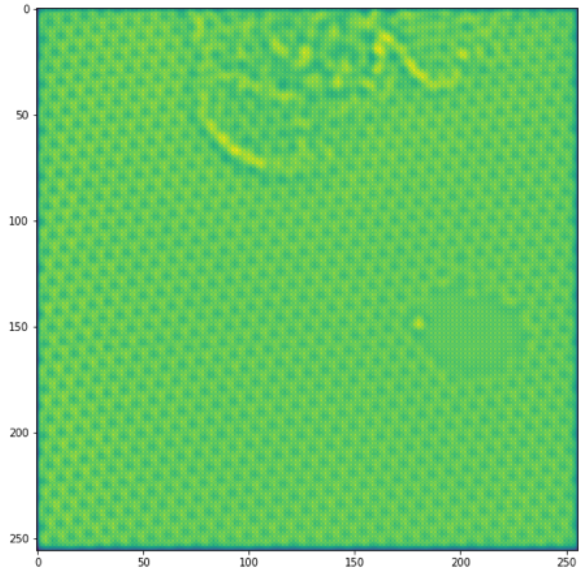
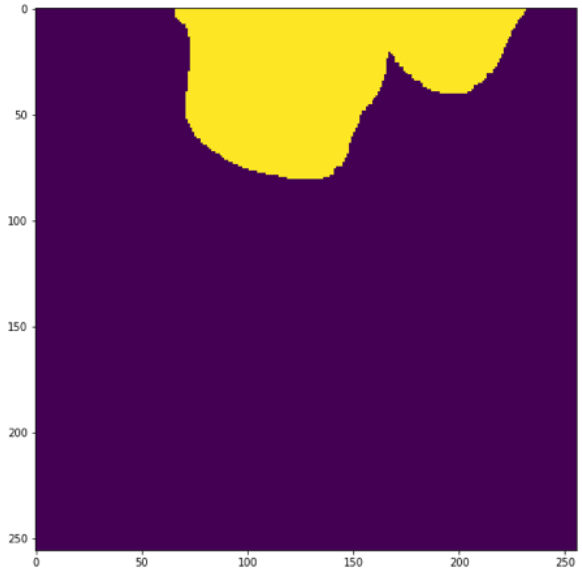
In [ ]:
```
X_train = [D]
y_train = [L]
for _ in range(100):
    for X, y in zip(X_train,y_train):

        X_=np.copy(X)
        y_=np.copy(y)


        print(X.shape)
        print(y.shape)



        X = X.reshape([1,1,X.shape[0],X.shape[1]])
        y = y.reshape([1,y.shape[0],y.shape[1]])

        X = torch.tensor(X)
        y = torch.tensor(y)
        X = X.to(device)  # [N, 1, H, W]
        y = y.to(device)  # [N, H, W] with class indices (0, 1)
        prediction = model(X)  # [N, 2, H, W]
```

```
        loss = F.cross_entropy(prediction, y)


        optim.zero_grad()
        loss.backward()
        optim.step()


        out = prediction.detach().numpy()

        #out = out.reshape([out.shape[3],out.shape[2],out.shape[1]])
        out = out[0,0,:,:]

        print(out.shape)
        if _%2==0:

            print('input image, ground truth')
            plt.figure(figsize = (20,20))
            plt.subplot(1,2,1)
            plt.imshow(X_)
            plt.subplot(1,2,2)
            plt.imshow(y_)
            plt.show()

            print('Unet output, ground truth')
            plt.figure(figsize = (20,20))
            plt.subplot(1,2,1)
            plt.imshow(out)
            plt.subplot(1,2,2)
            plt.imshow(y_)
            plt.show()
        print('pass')
```

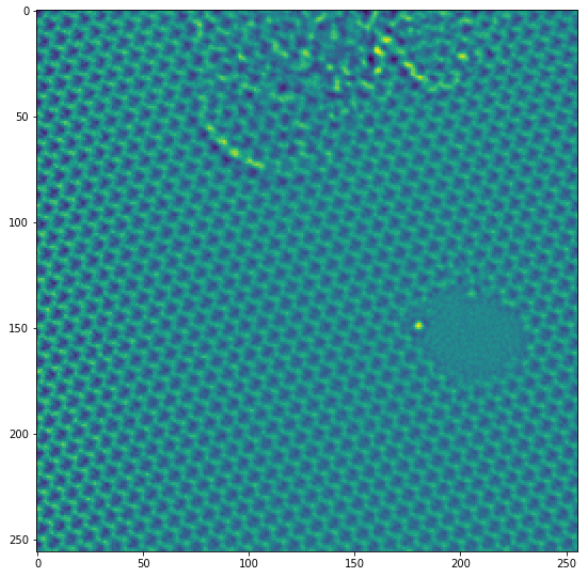
```
(256, 256)
(256, 256)
(256, 256)
input image, ground truth
```

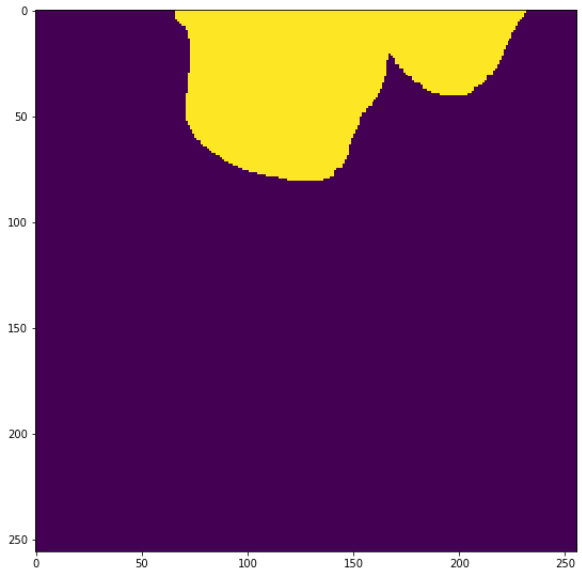



Unet output, ground truth

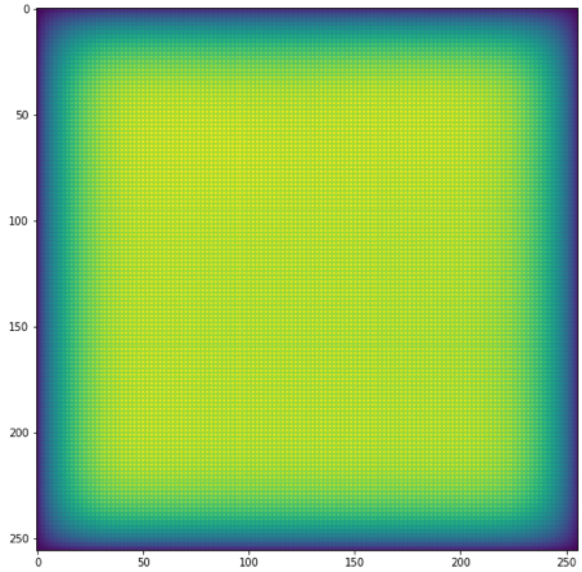



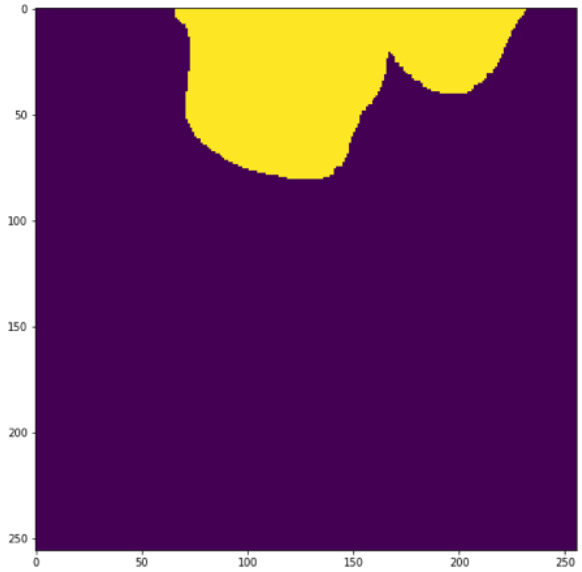
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth



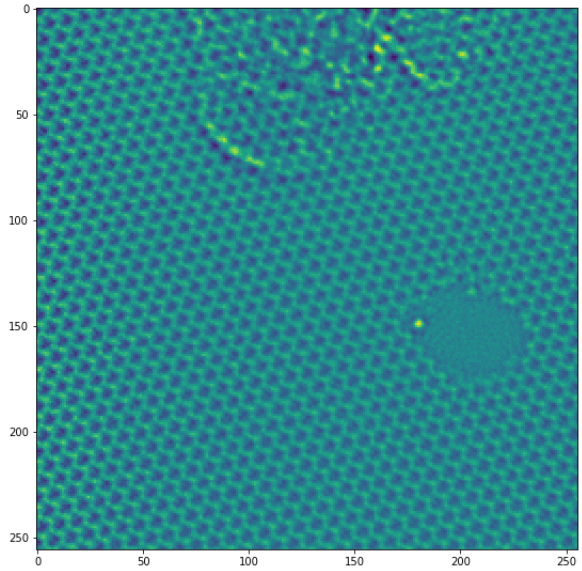


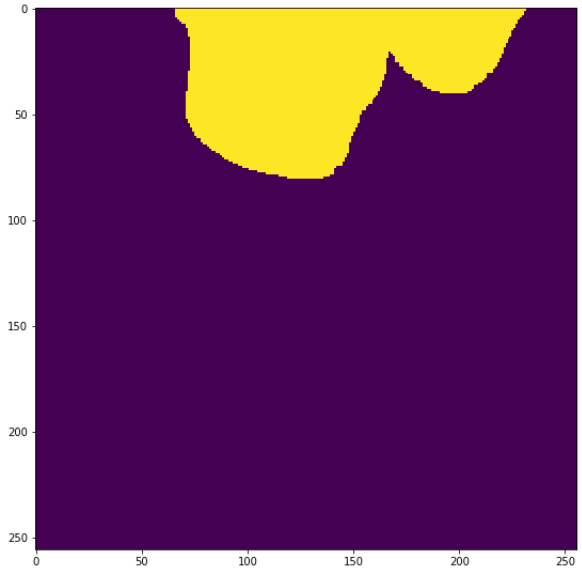
Unet output, ground truth



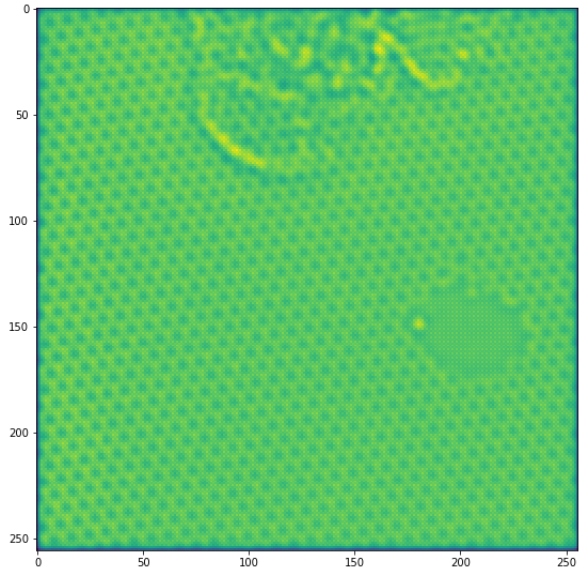


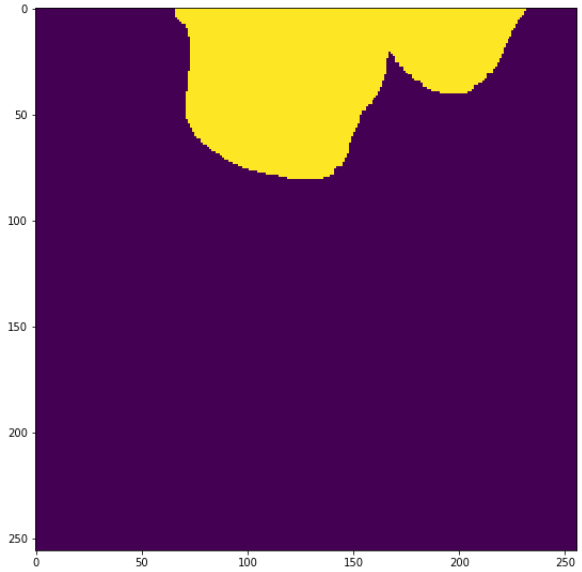
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth



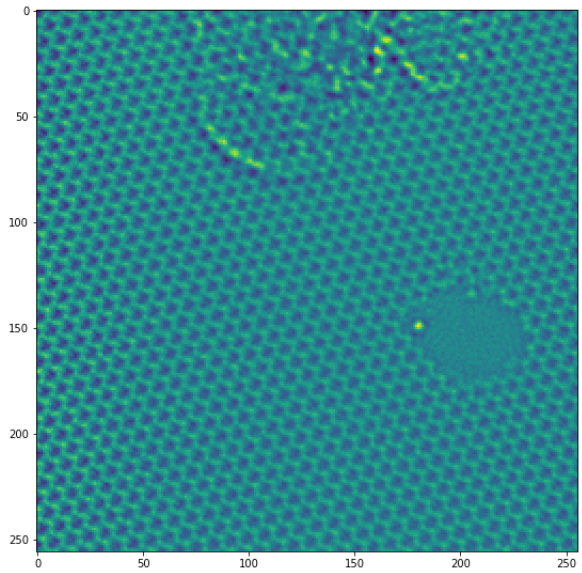
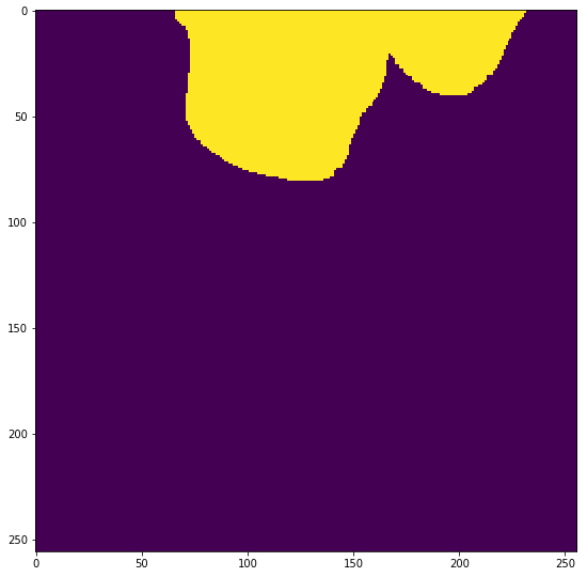


Unet output, ground truth

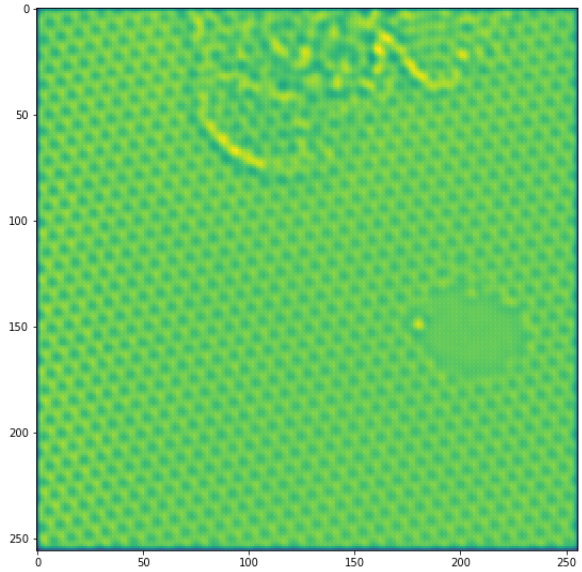
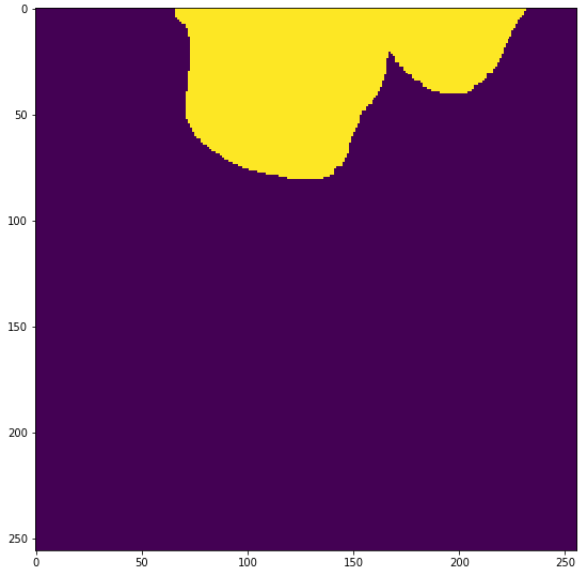




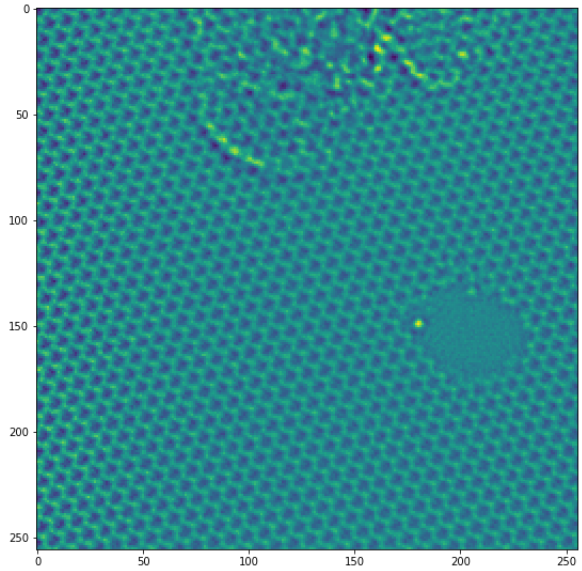
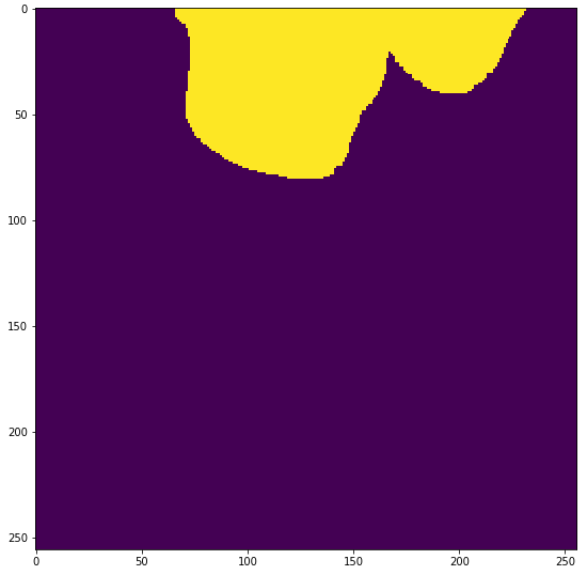
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth

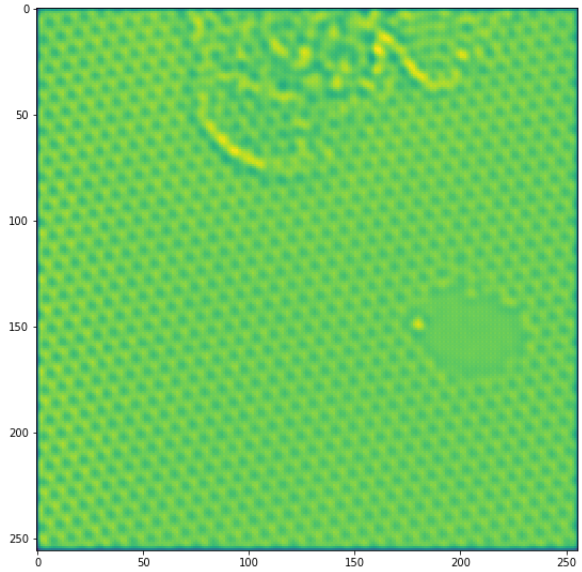
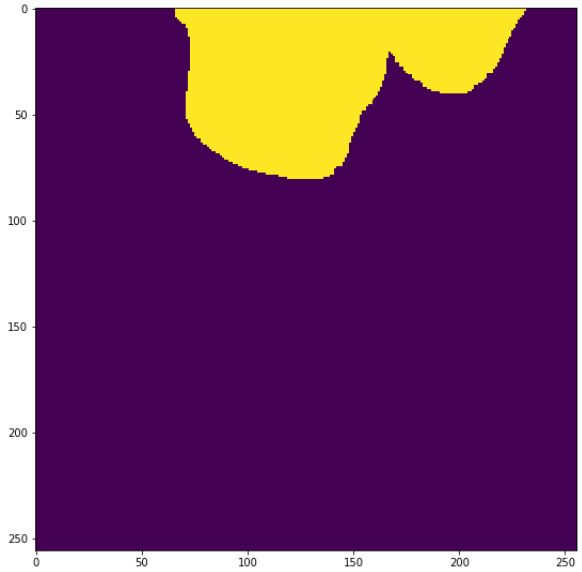
Unet output, ground truth

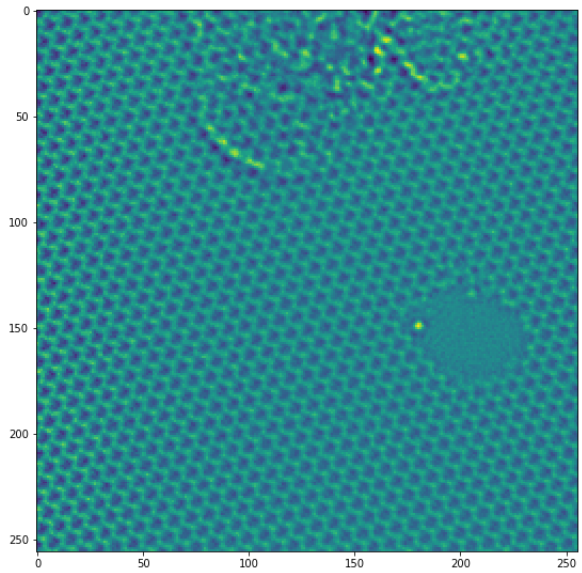
```
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth
```

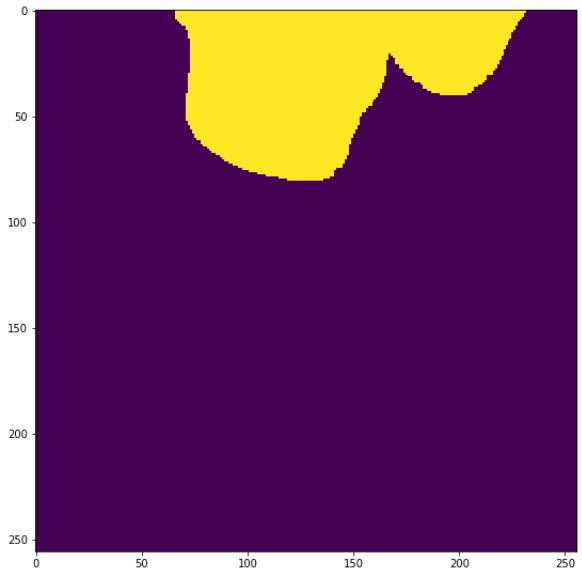



```
Unet output, ground truth
```

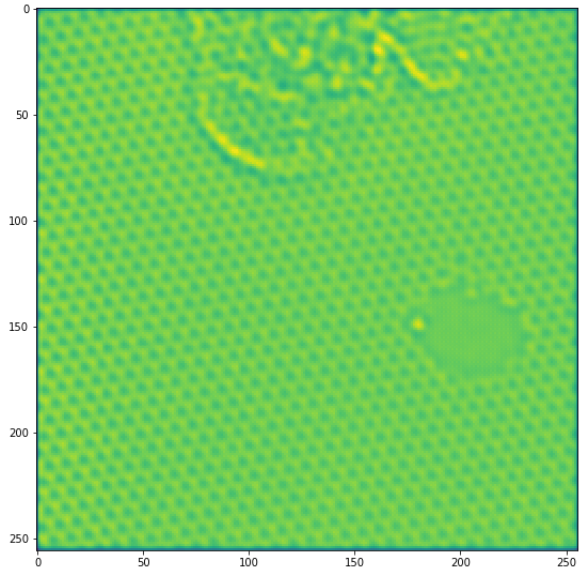



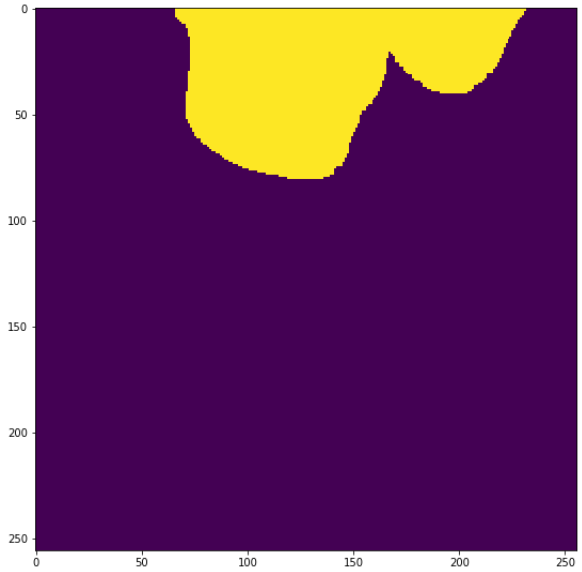
```
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth
```



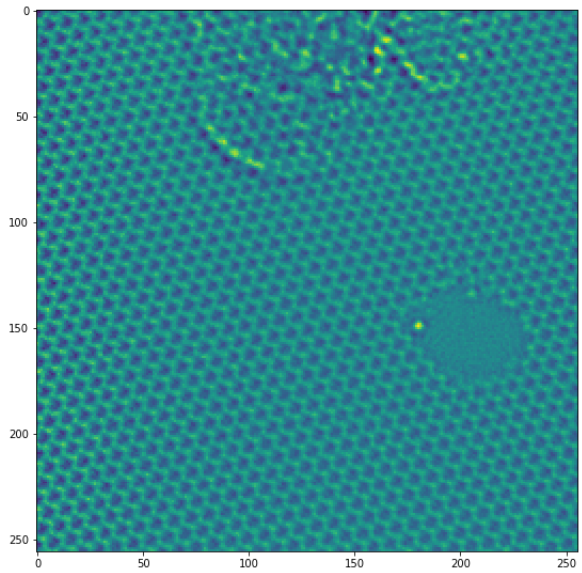
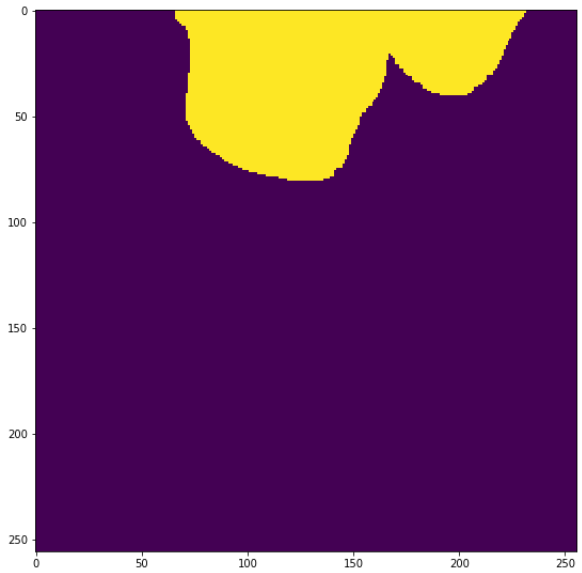


Unet output, ground truth

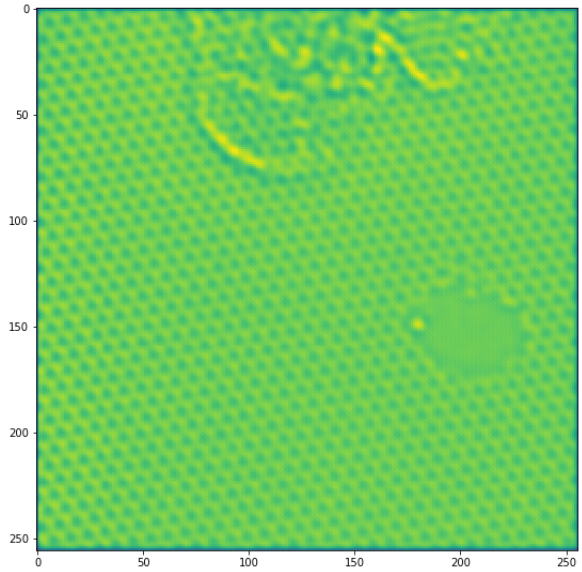
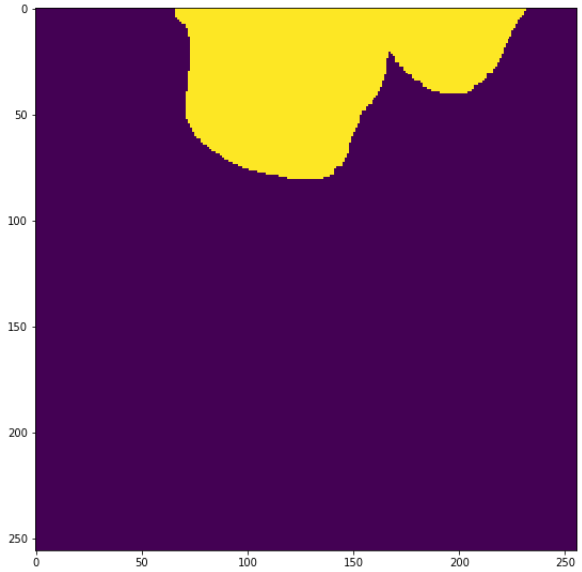




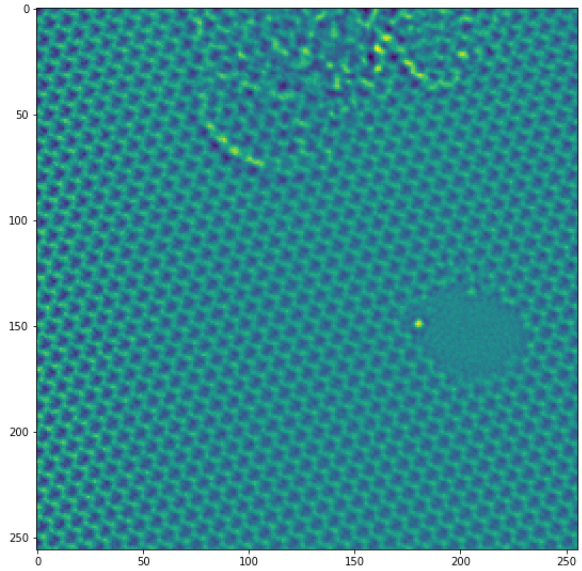
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth

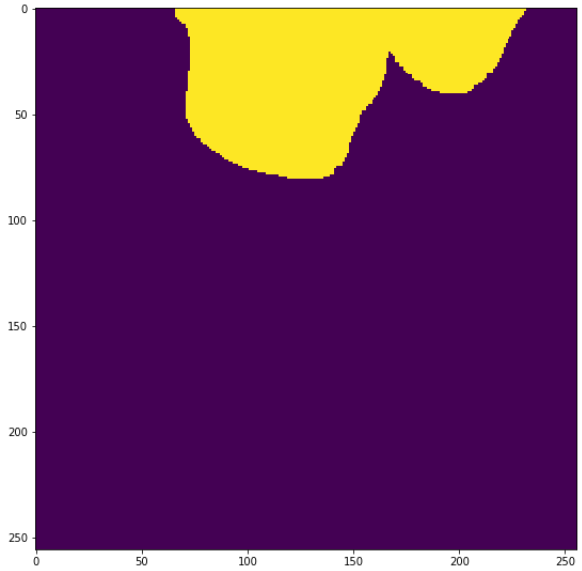



Unet output, ground truth

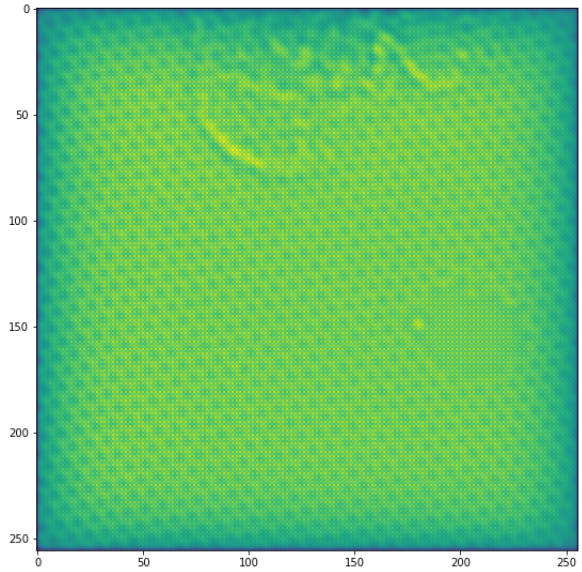



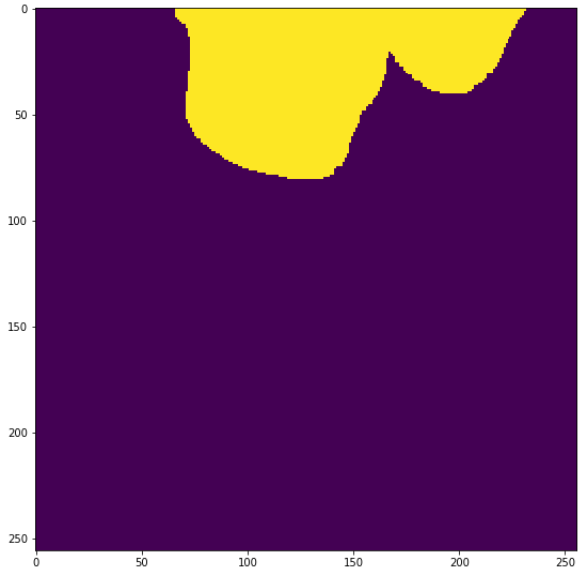
```
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth
```



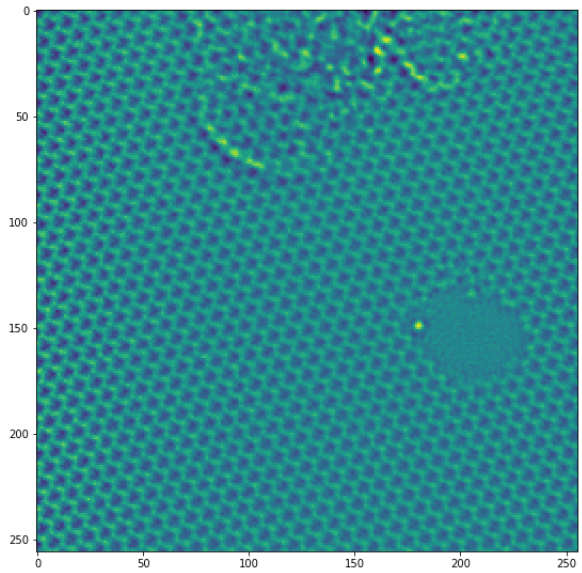


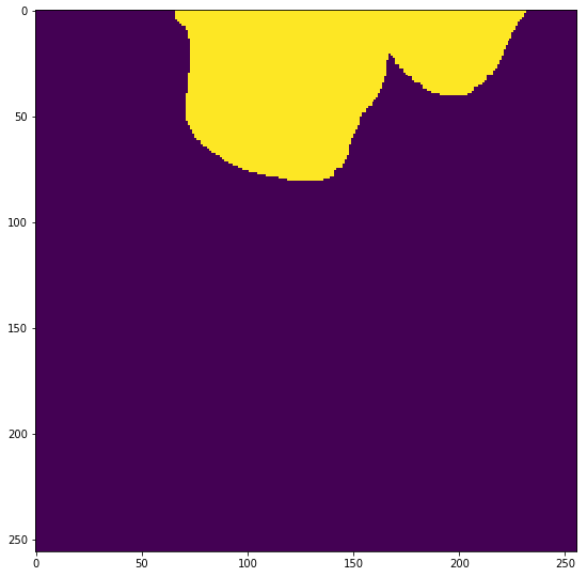
```
Unet output, ground truth
```



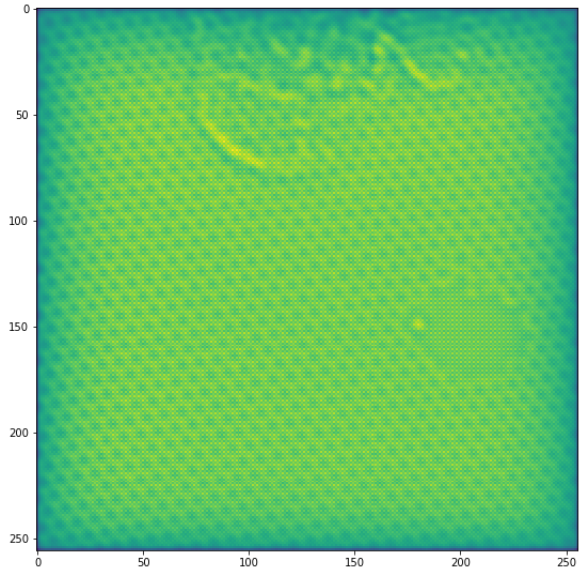


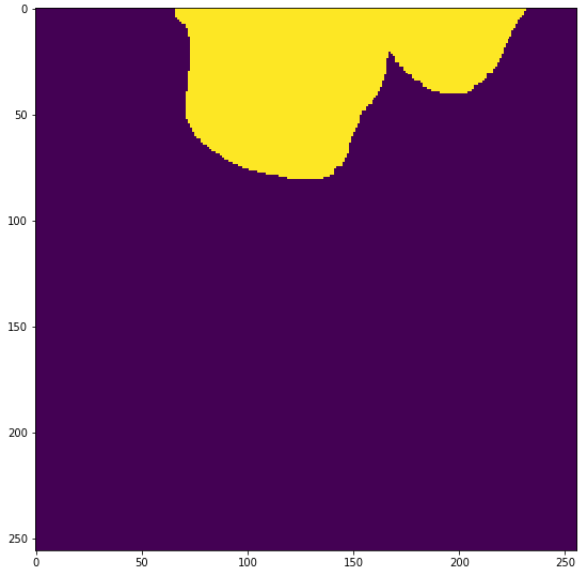
```
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth
```



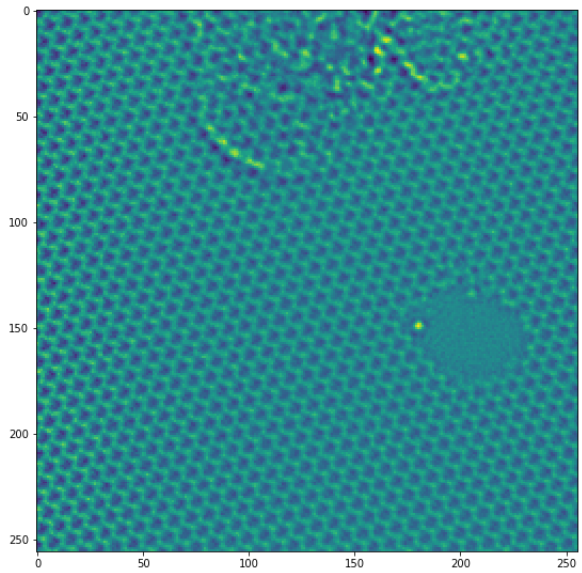


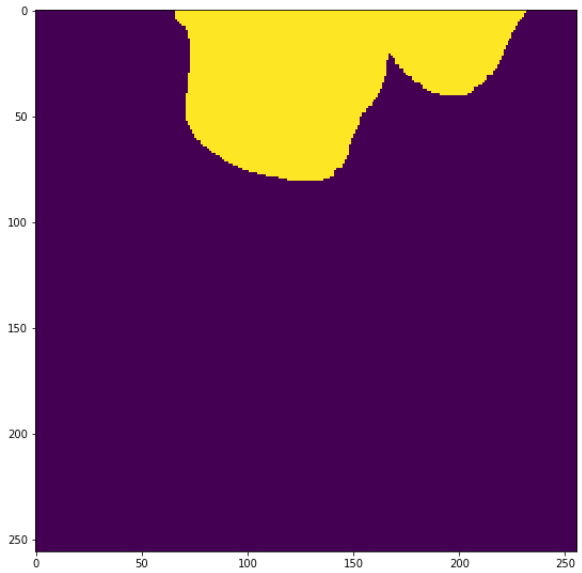
Unet output, ground truth



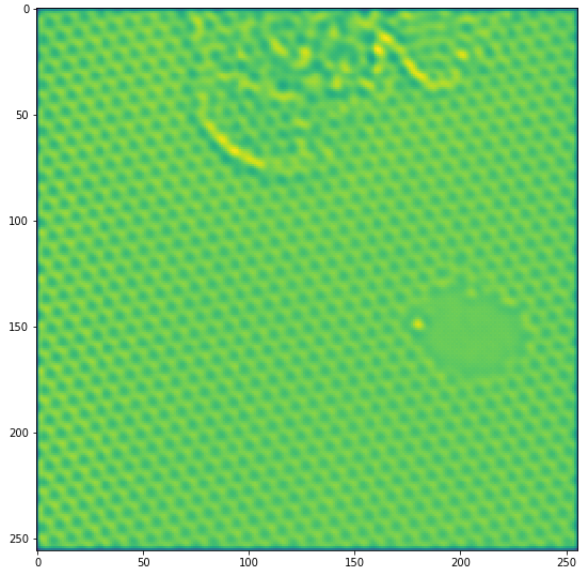


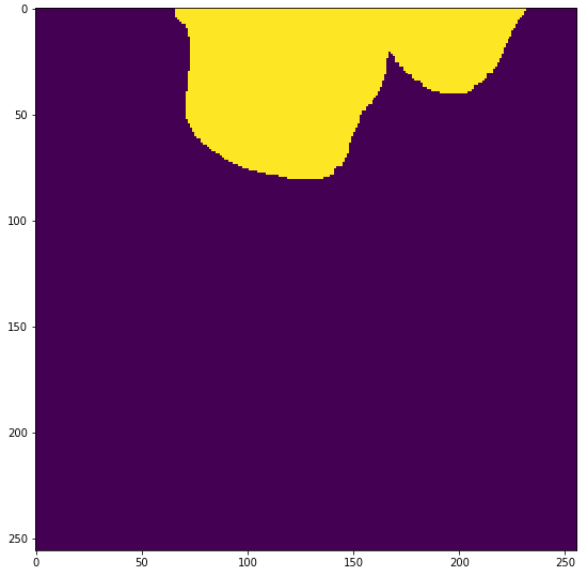
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth



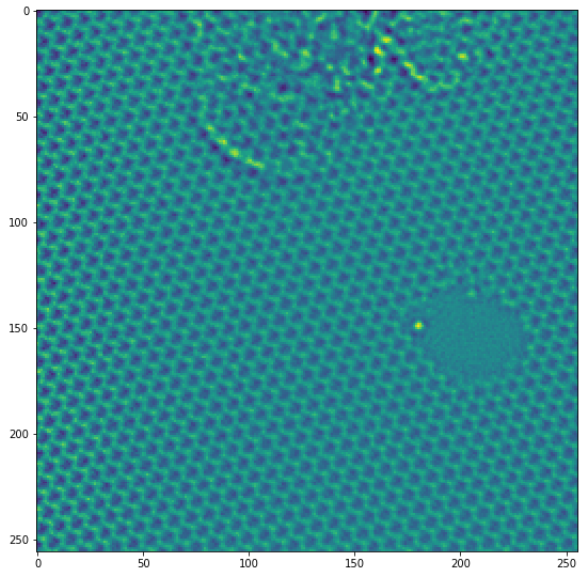
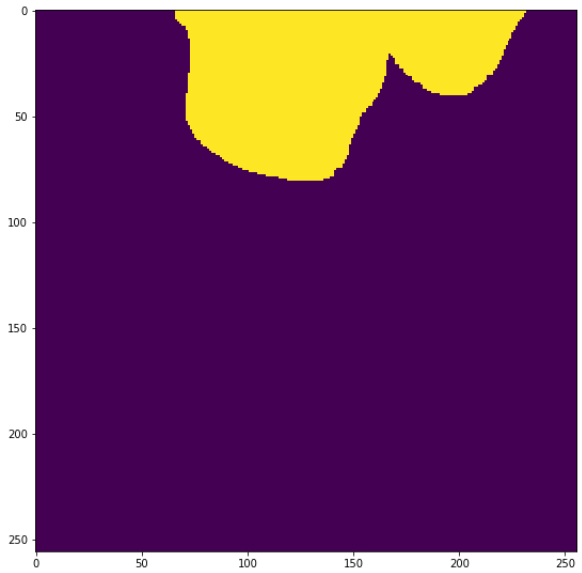


Unet output, ground truth

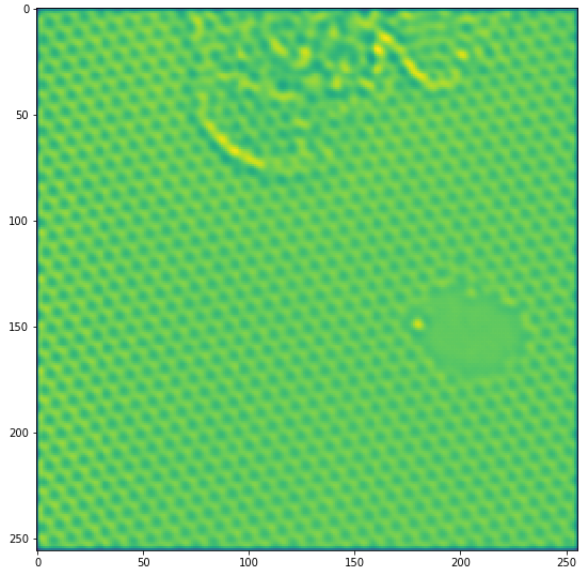
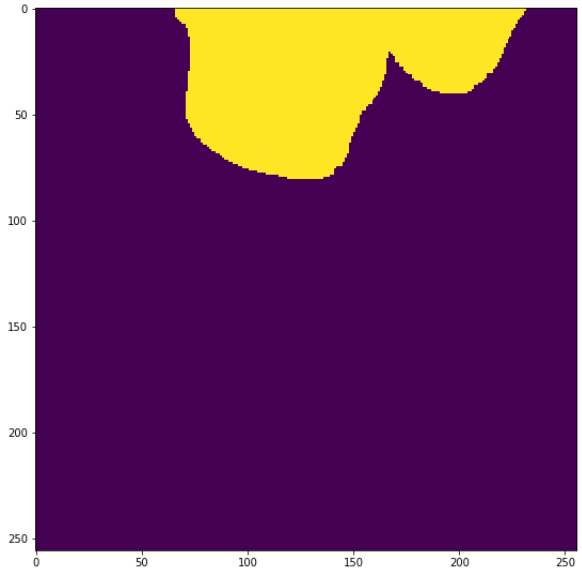




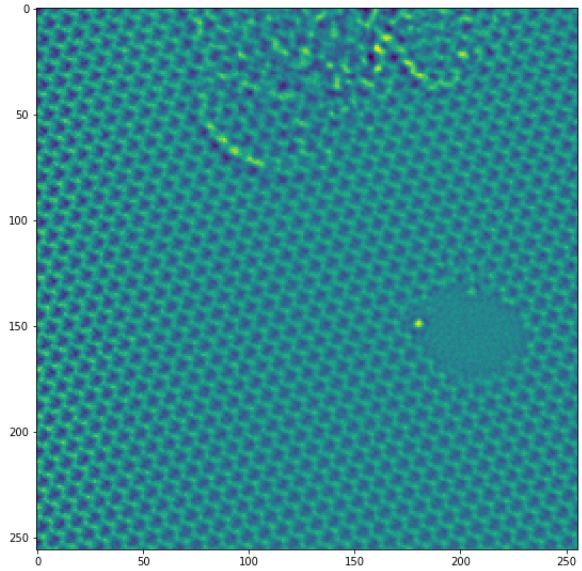
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth

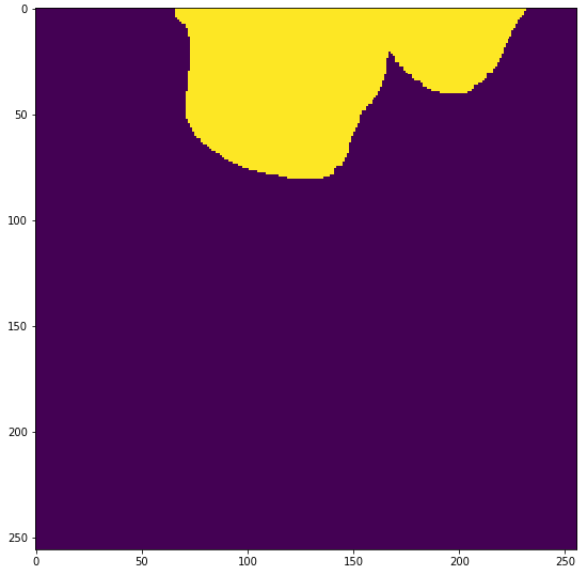



Unet output, ground truth

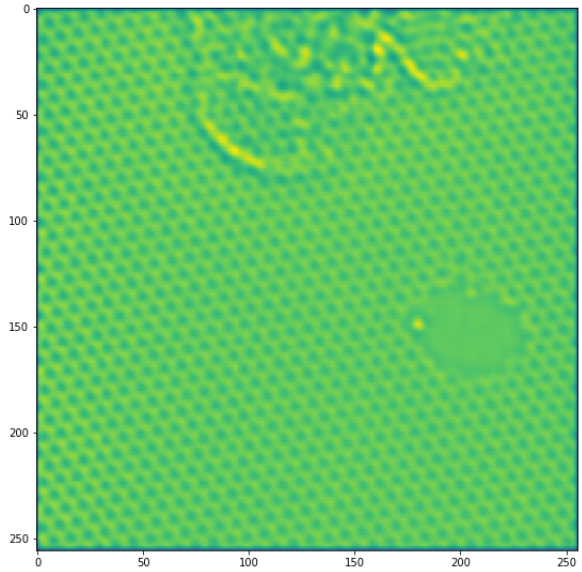



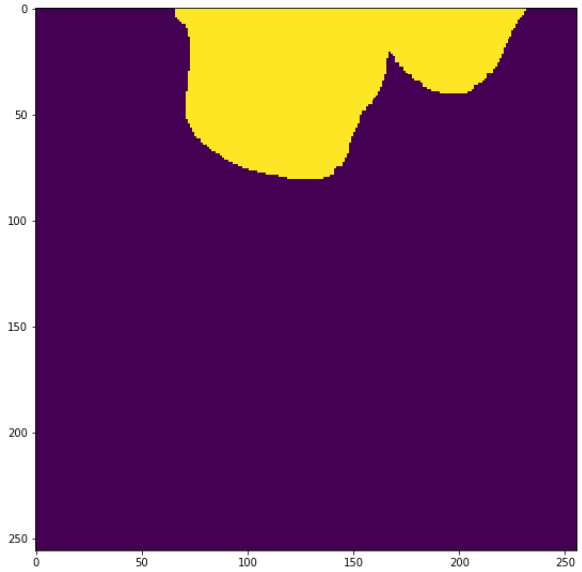
```
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth
```



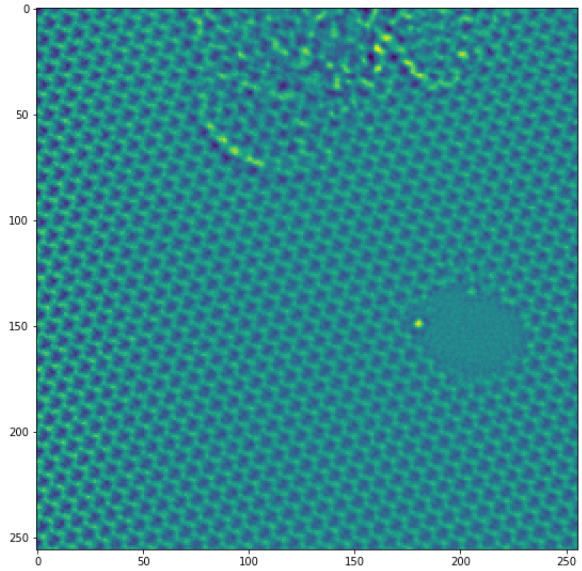
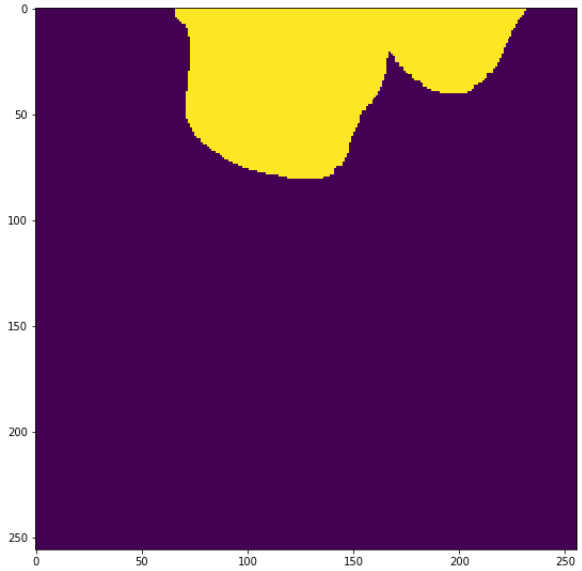


Unet output, ground truth

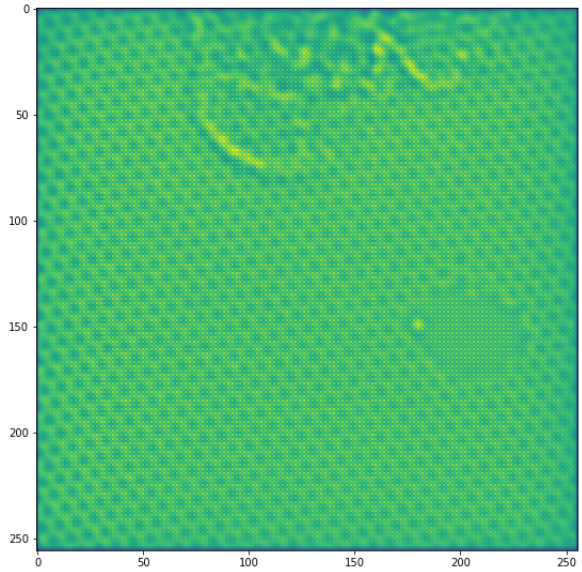
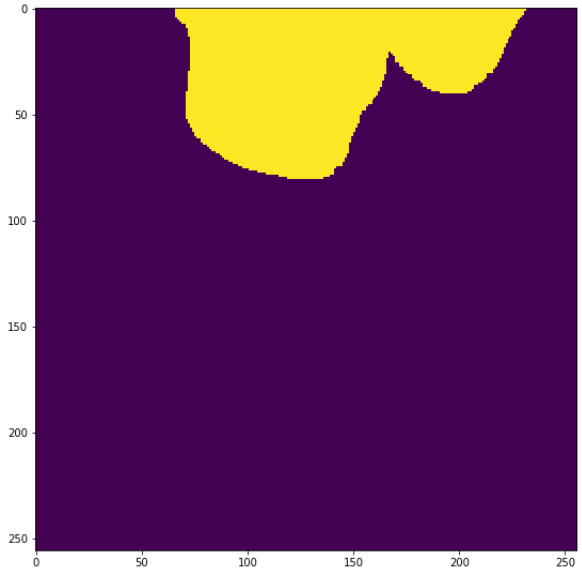




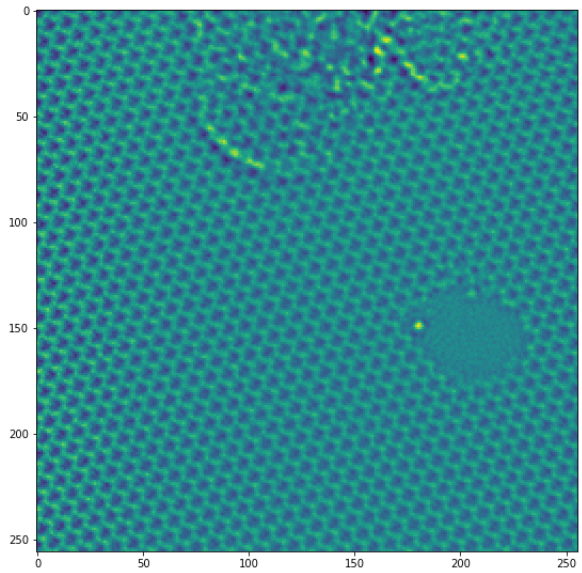
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth

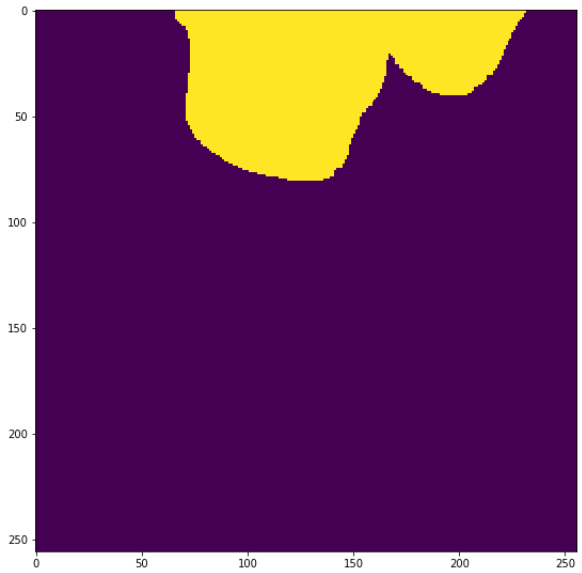



Unet output, ground truth

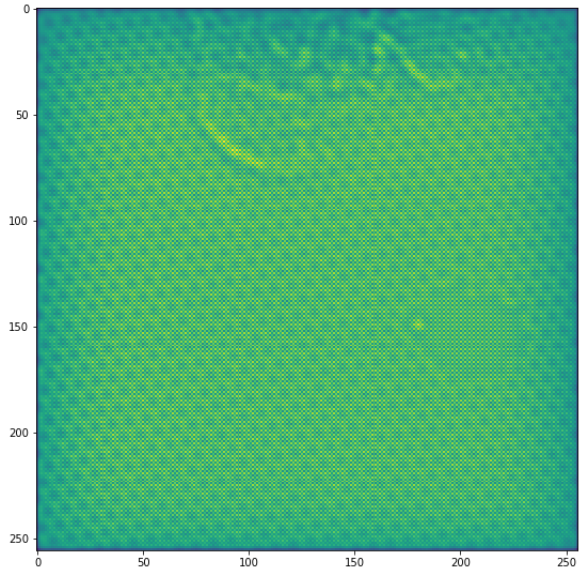



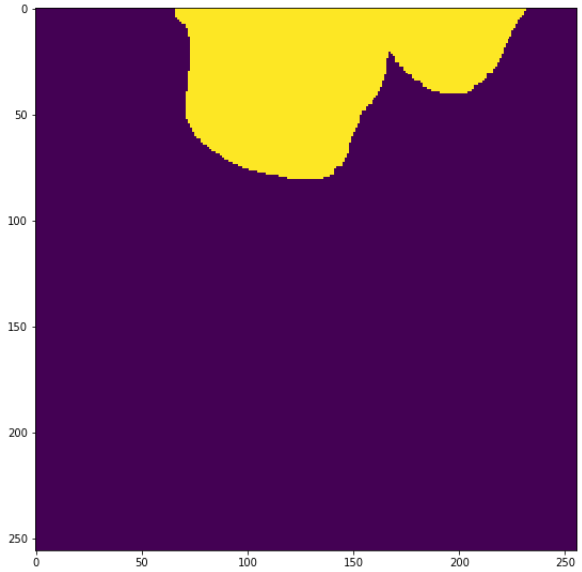
```
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth
```



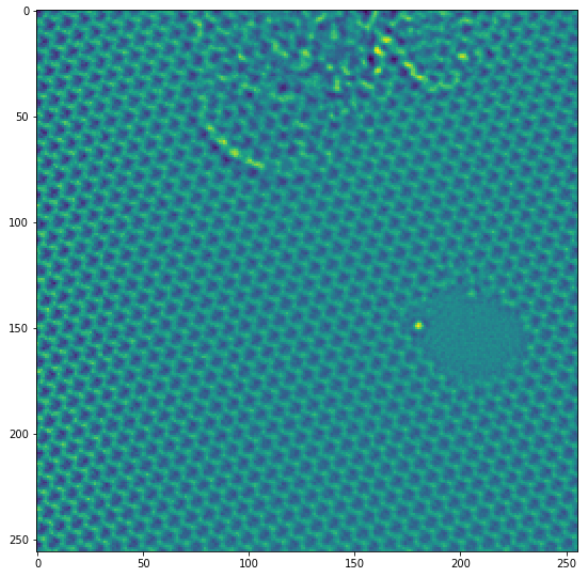


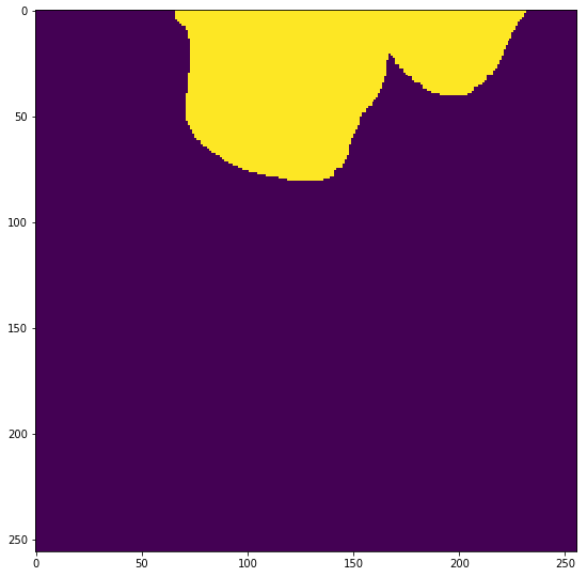
Unet output, ground truth



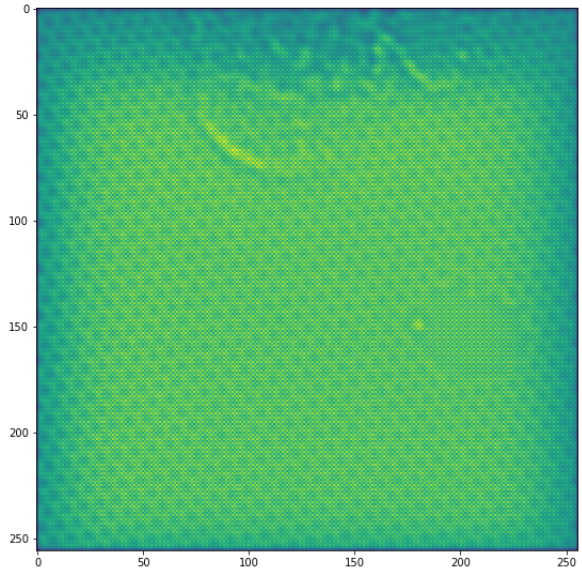


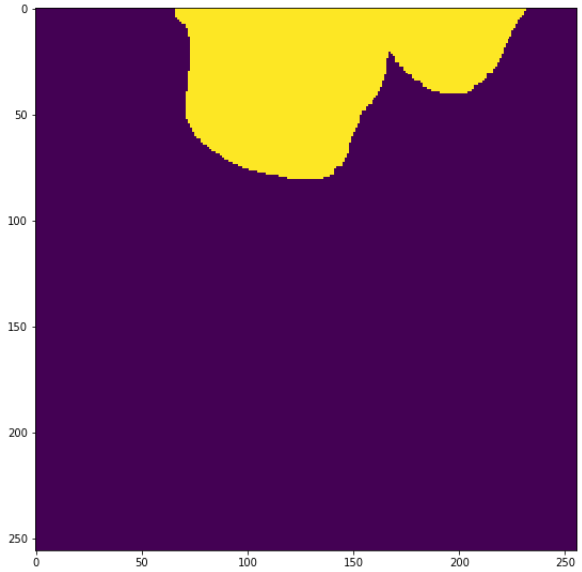
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth



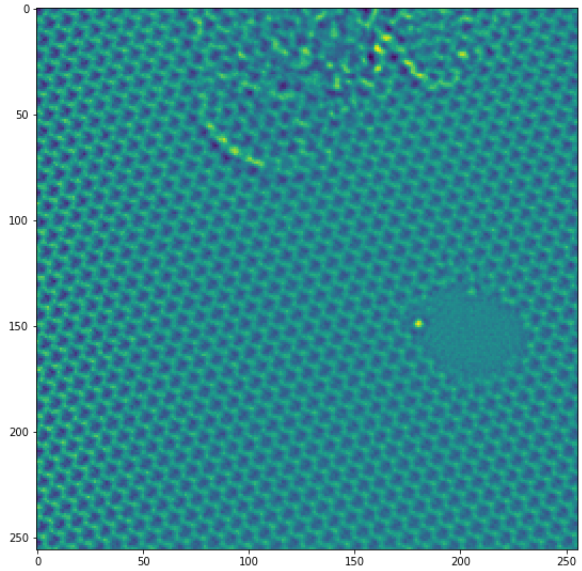


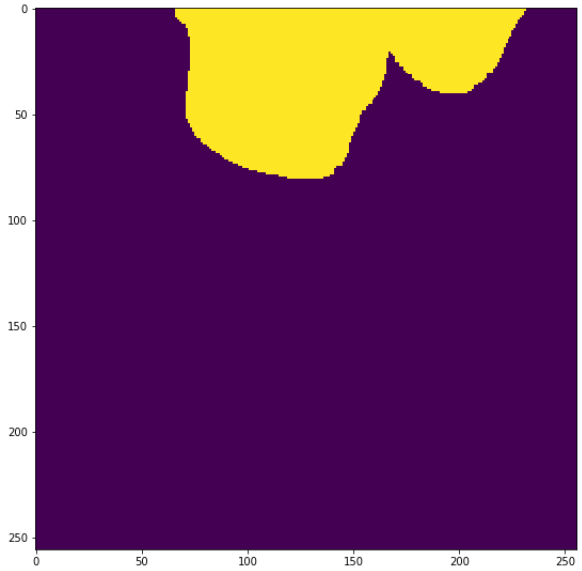
Unet output, ground truth



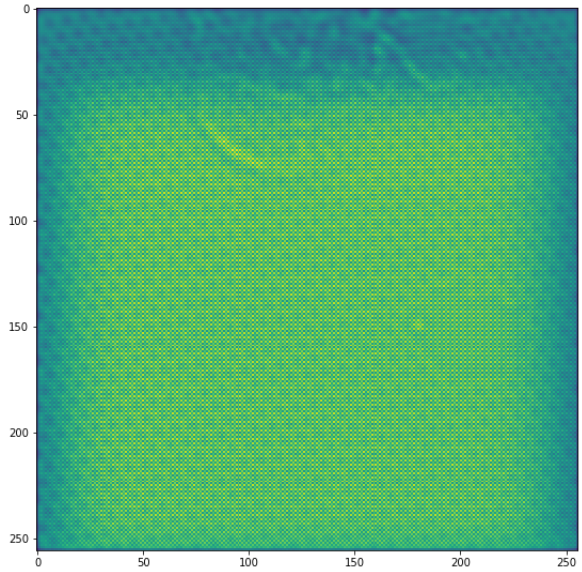


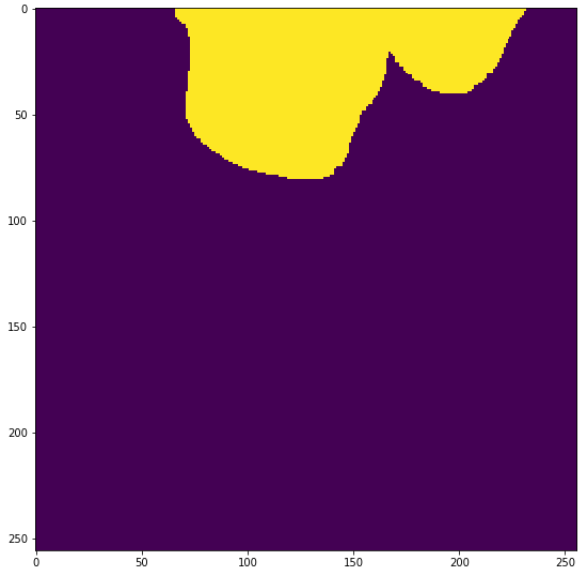
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth



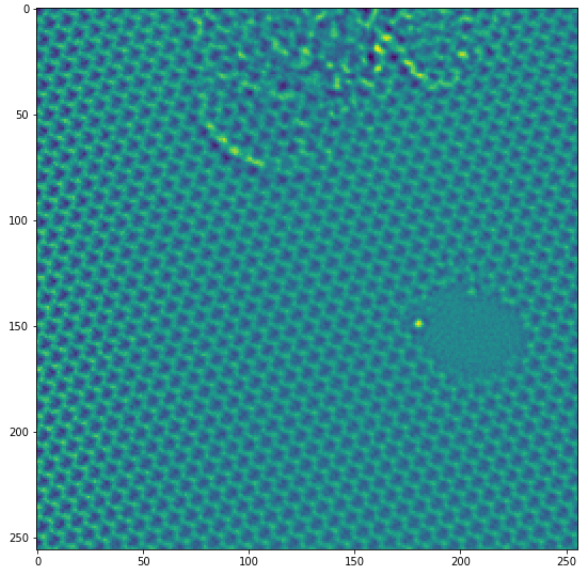


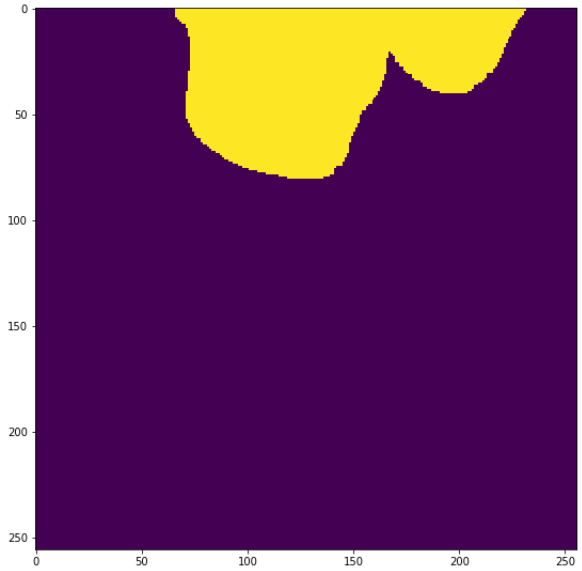
Unet output, ground truth



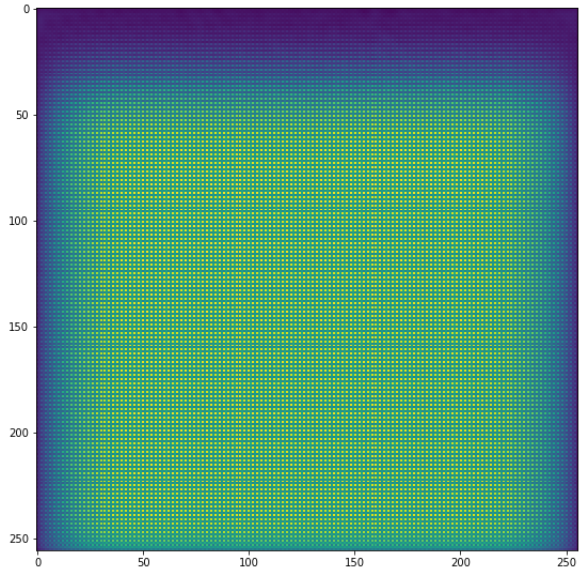


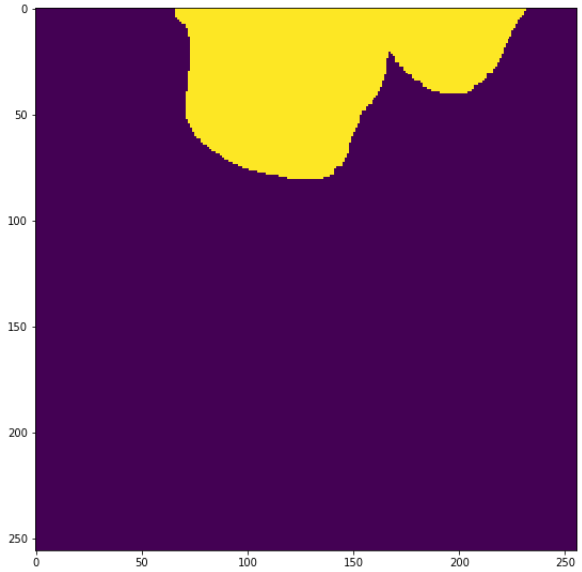
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth



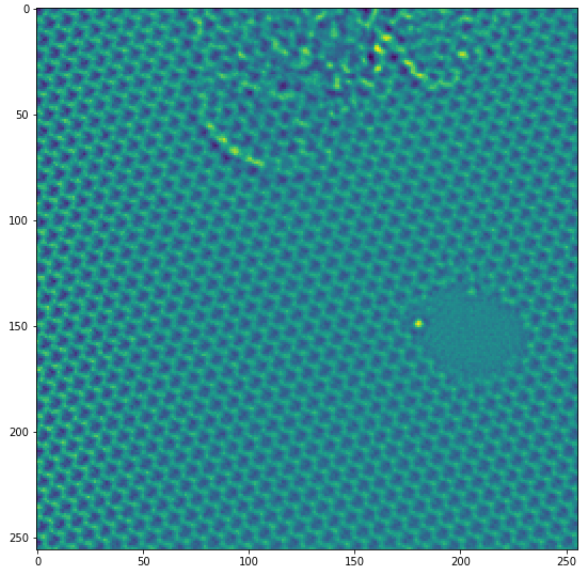
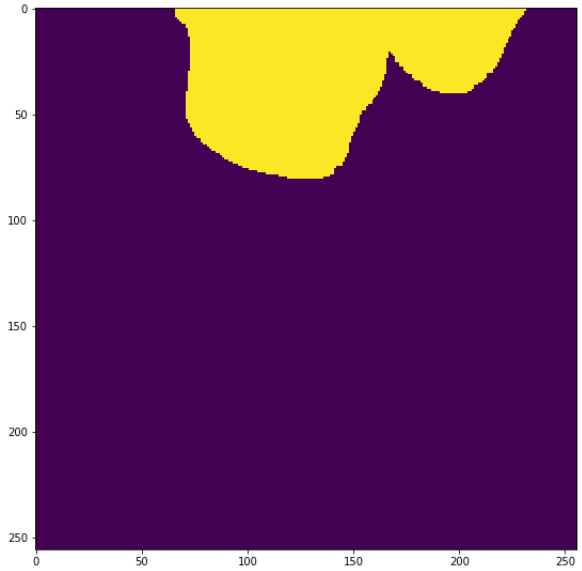


Unet output, ground truth

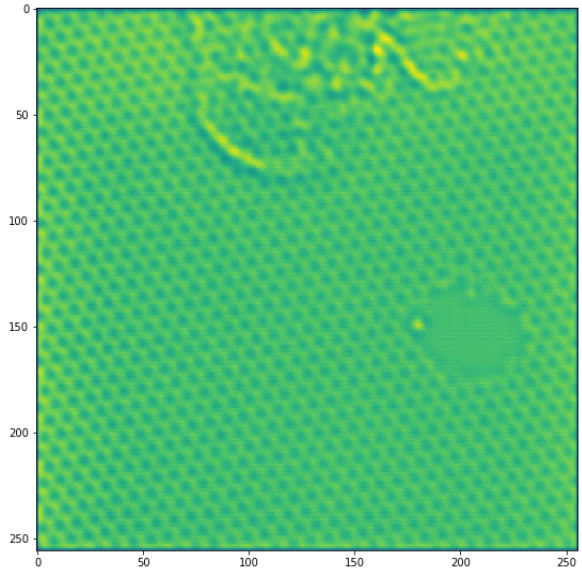
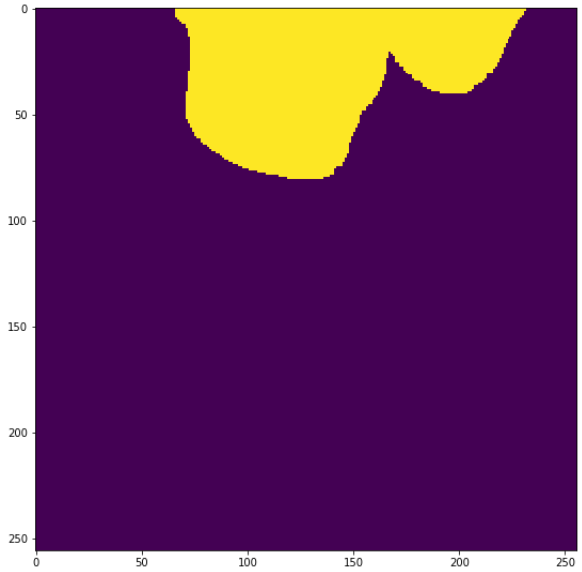




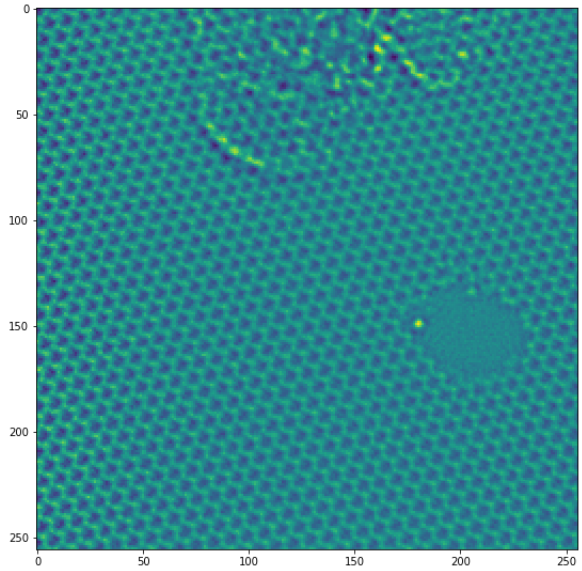
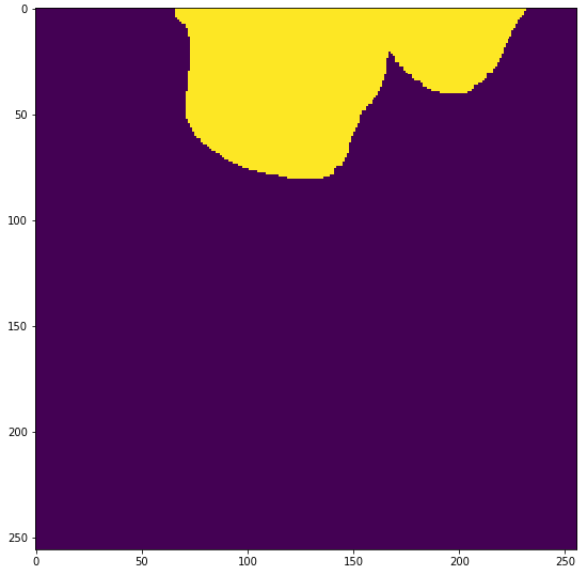
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth

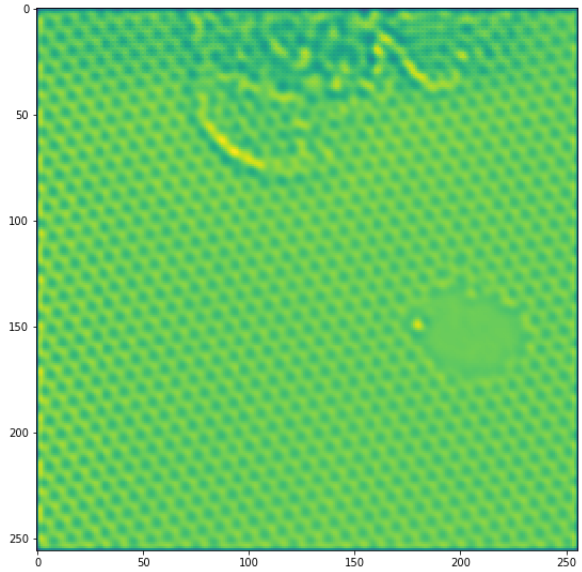
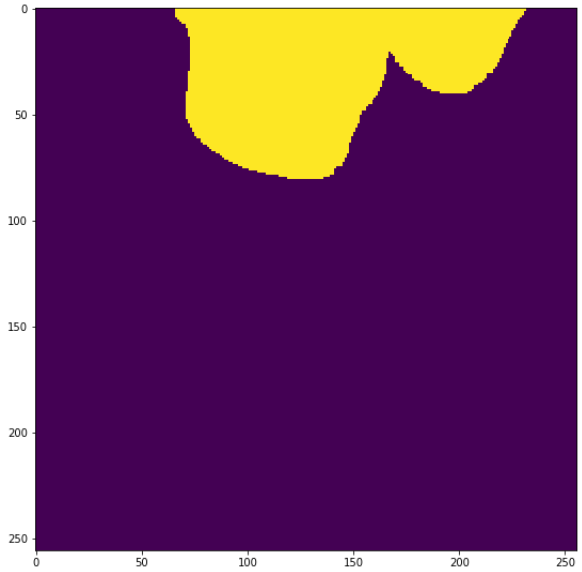
Unet output, ground truth

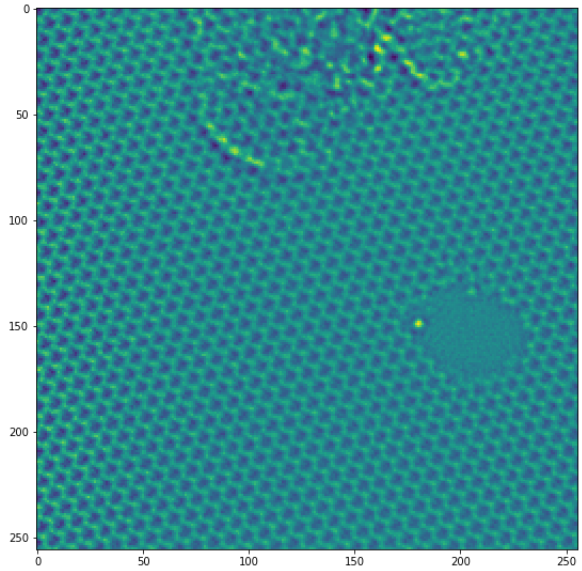
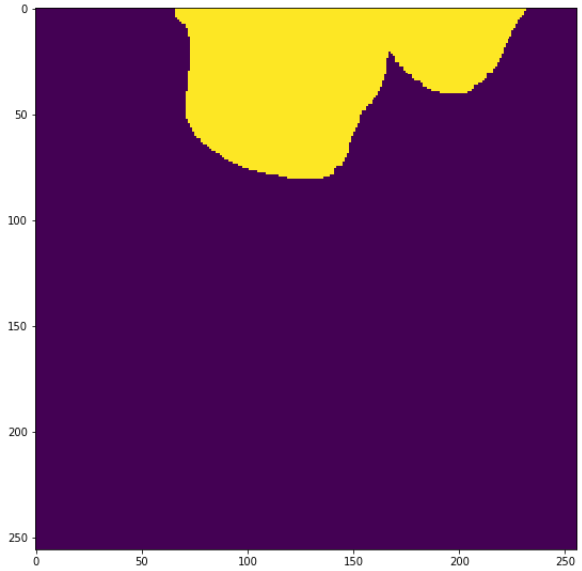
```
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth
```

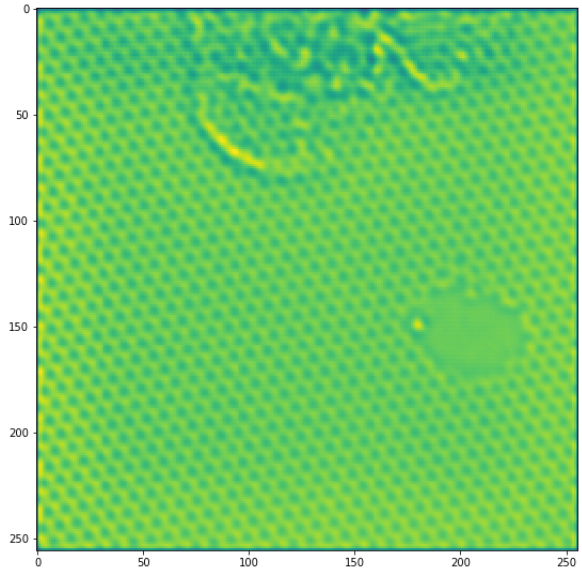
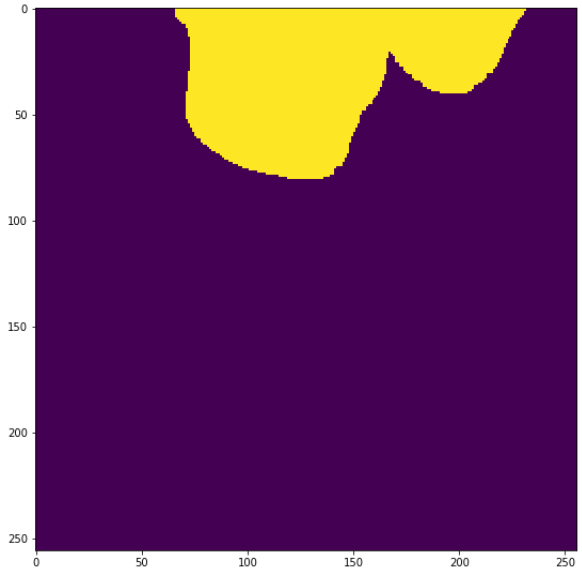
Unet output, ground truth

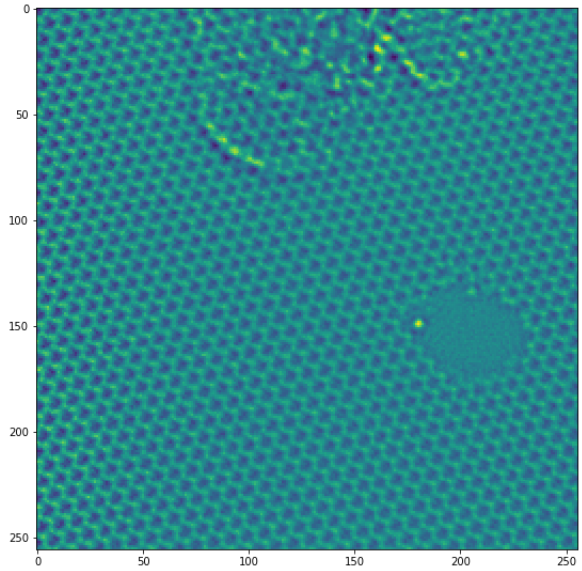
```
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth
```

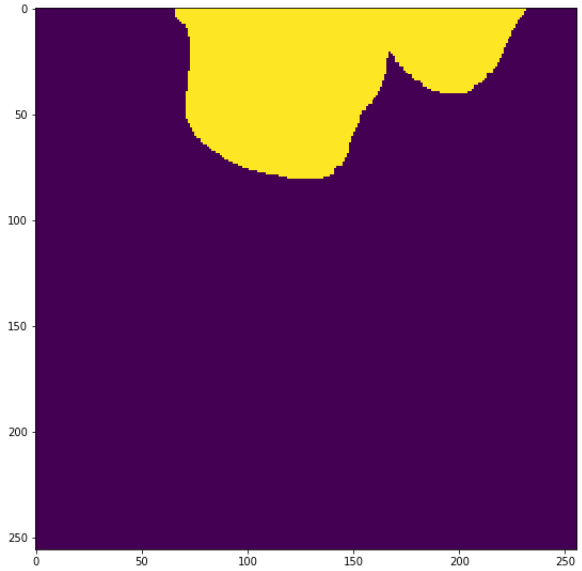



Unet output, ground truth

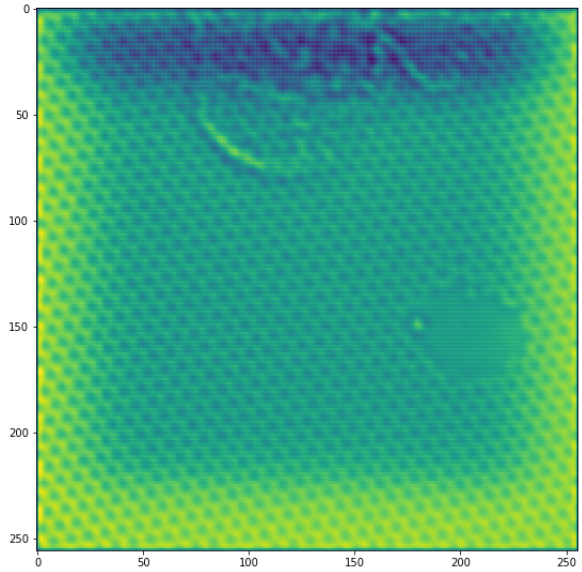



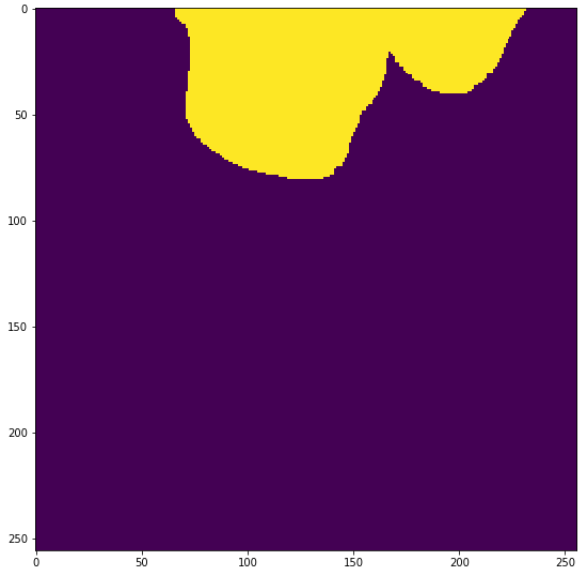
```
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth
```



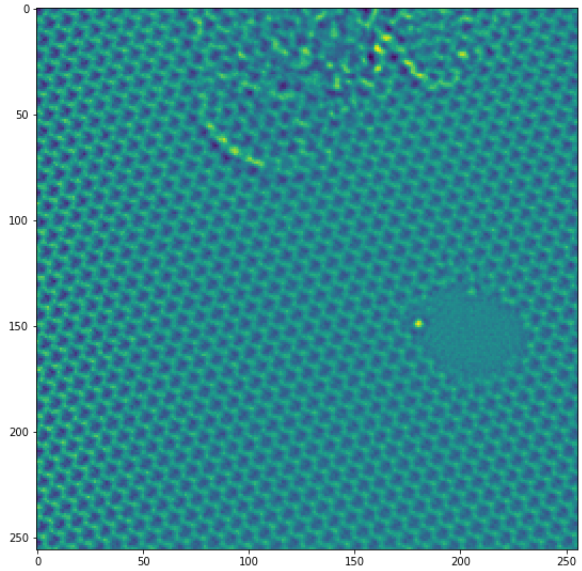
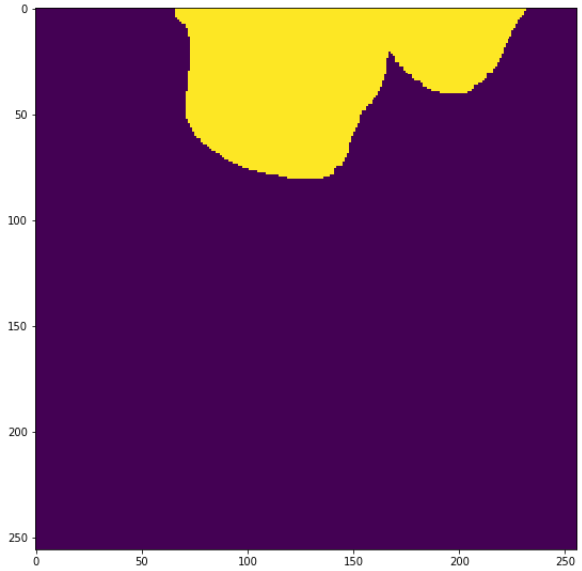


Unet output, ground truth

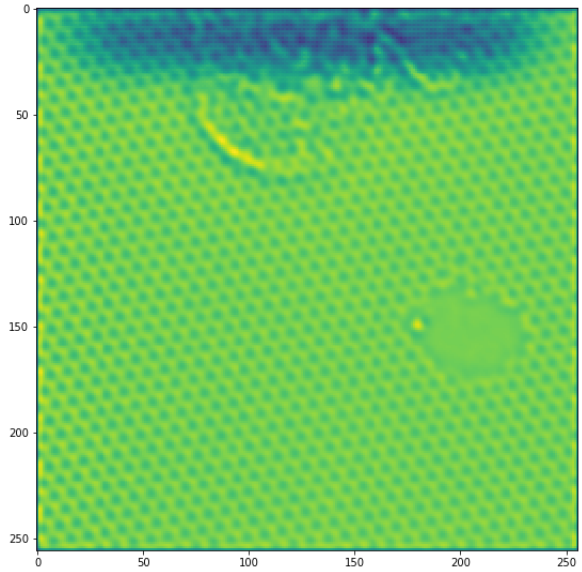
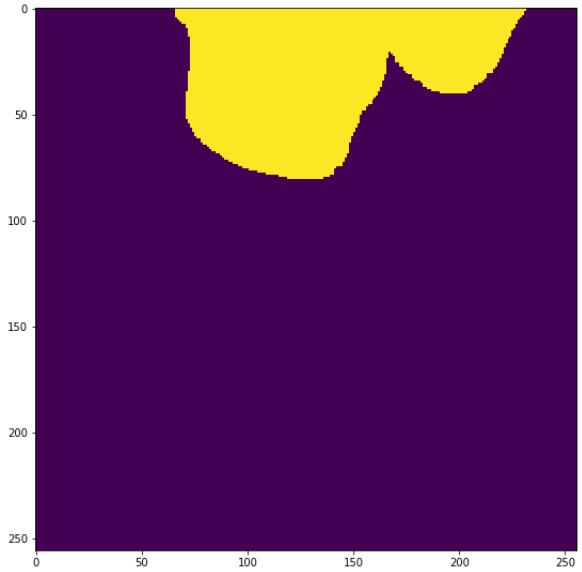




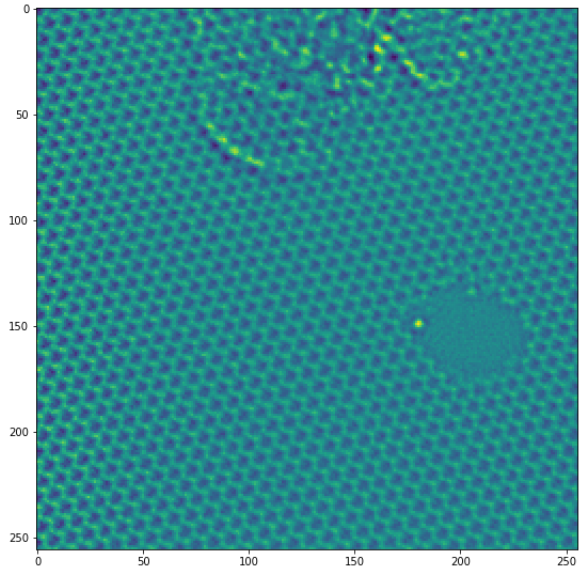
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth

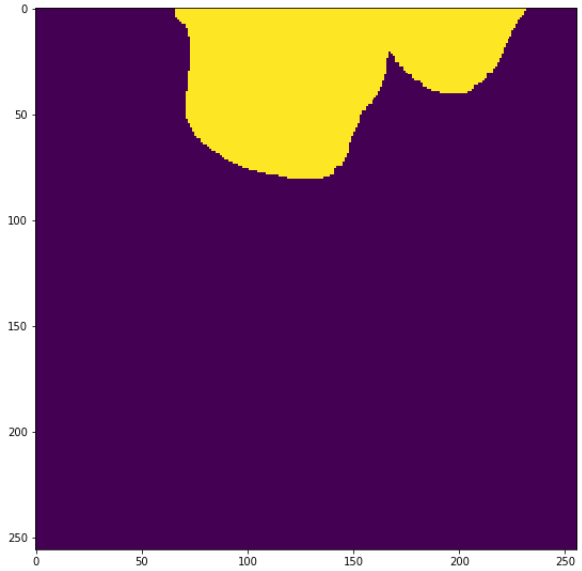



Unet output, ground truth

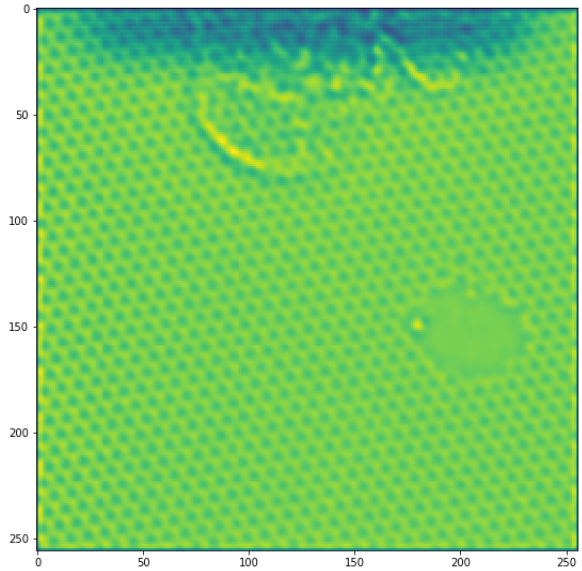



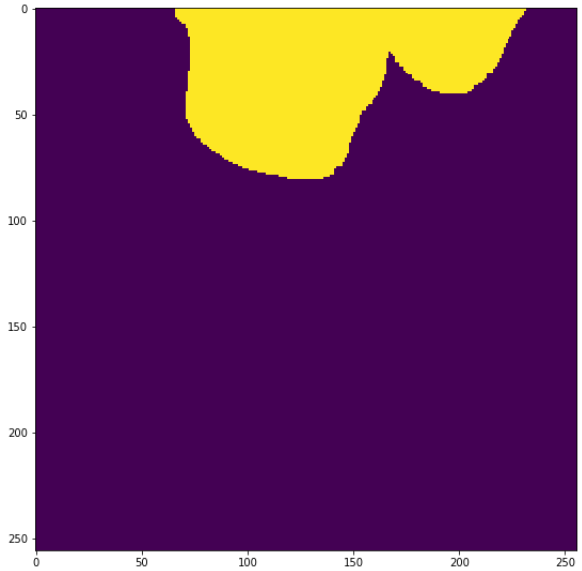
pass
(256, 256)
(256, 256)
(256, 256)
pass
(256, 256)
(256, 256)
(256, 256)
input image, ground truth





```
Unet output, ground truth
```





```
pass
(256, 256)
(256, 256)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: