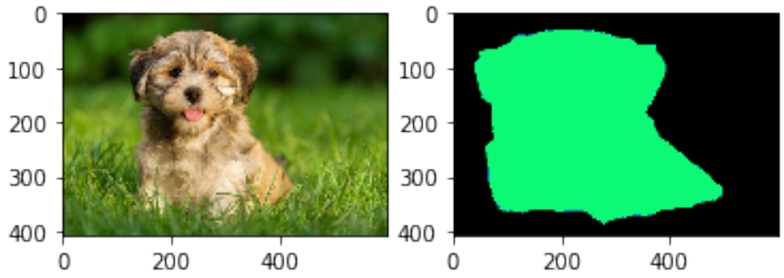
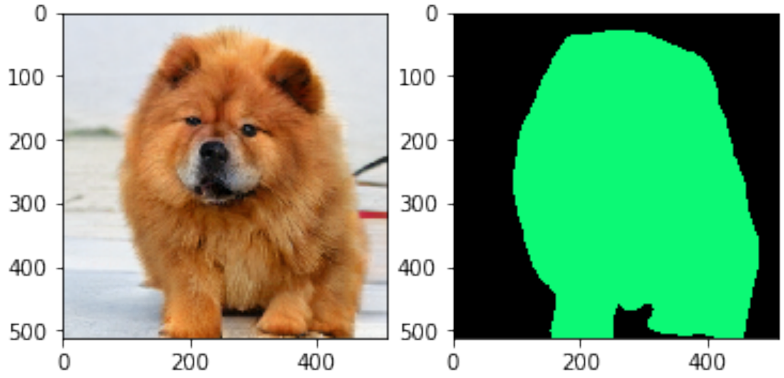
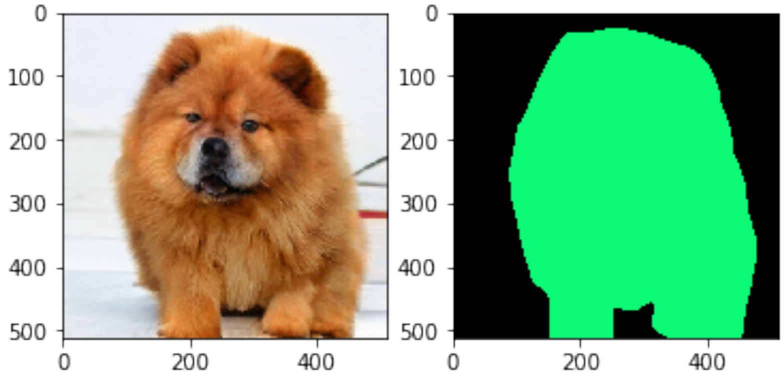


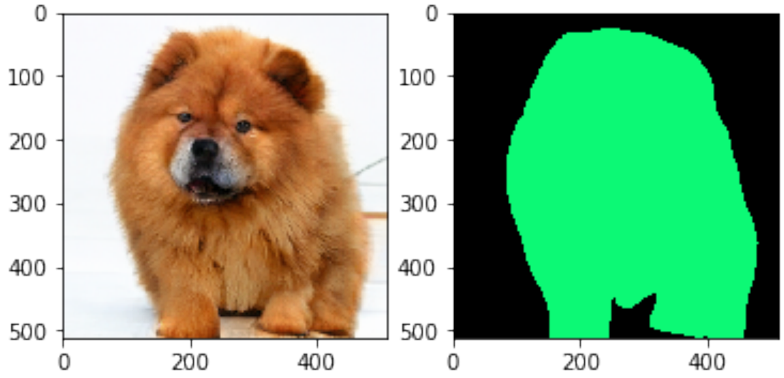
```
cost 0  is  tensor(1.3593, grad_fn=<NllLoss2DBackward>)
```



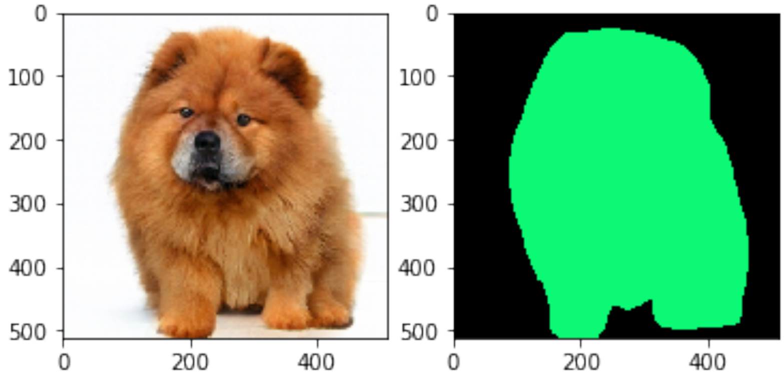
```
cost 1  is  tensor(1.3123, grad_fn=<NllLoss2DBackward>)
```



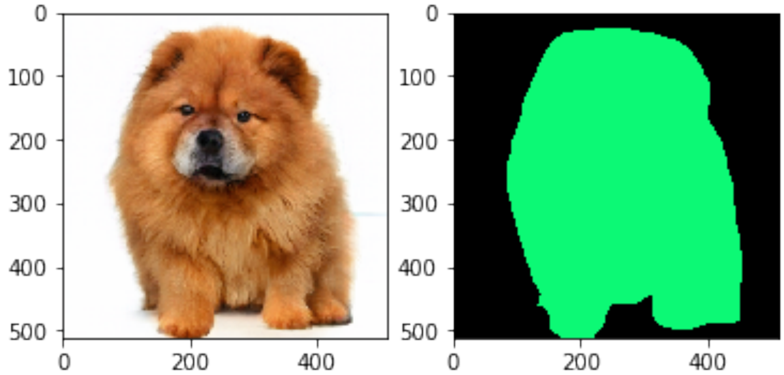
```
cost 2  is  tensor(0.9782, grad_fn=<NllLoss2DBackward>)
```



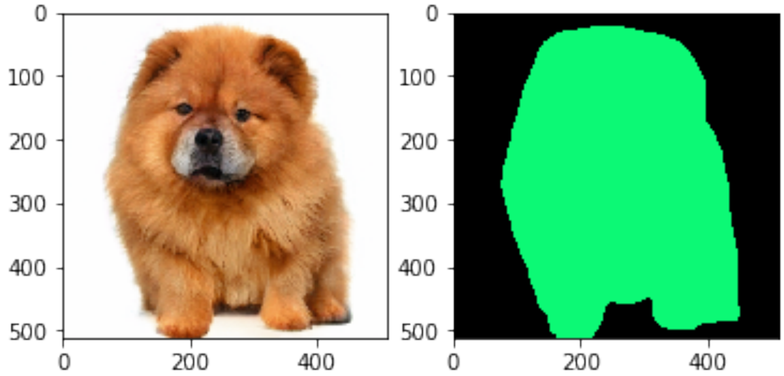
```
cost 3  is  tensor(1.1013, grad_fn=<NllLoss2DBackward>)
```



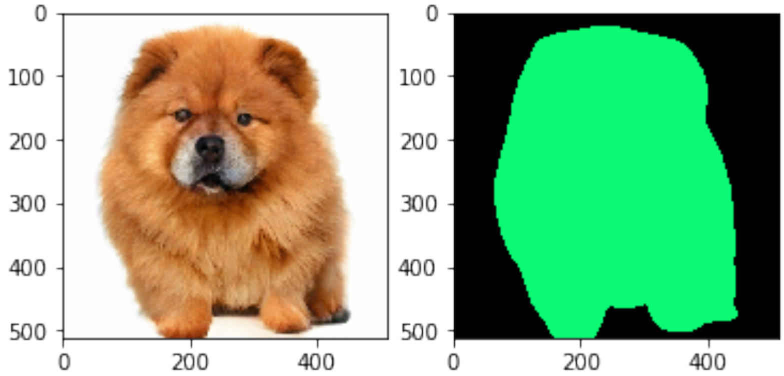
```
cost 4  is  tensor(1.0432, grad_fn=<NllLoss2DBackward>)
```



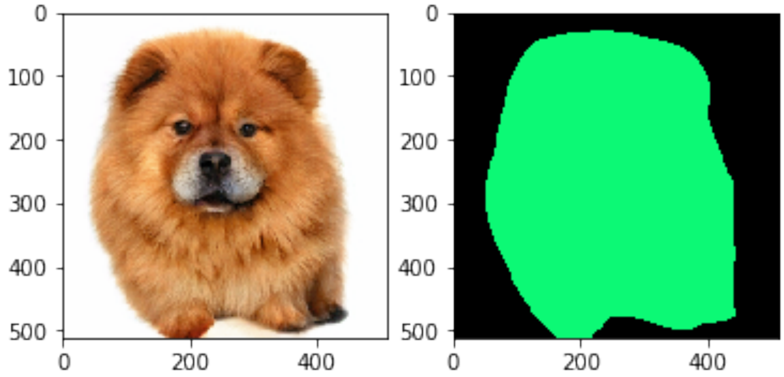
```
cost 5  is  tensor(0.8668, grad_fn=<NllLoss2DBackward>)
```



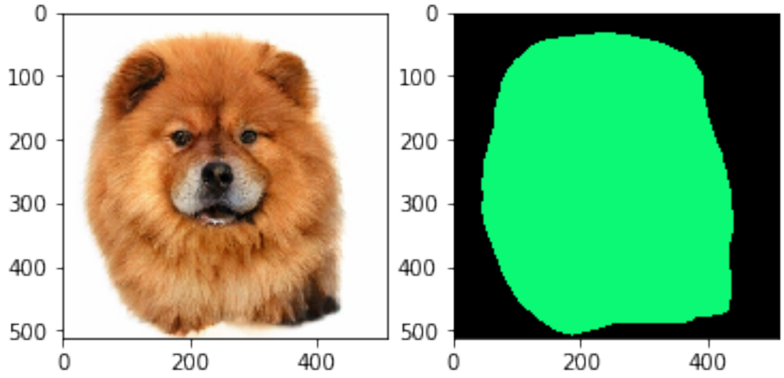
```
cost 6  is  tensor(0.7505, grad_fn=<NllLoss2DBackward>)
```



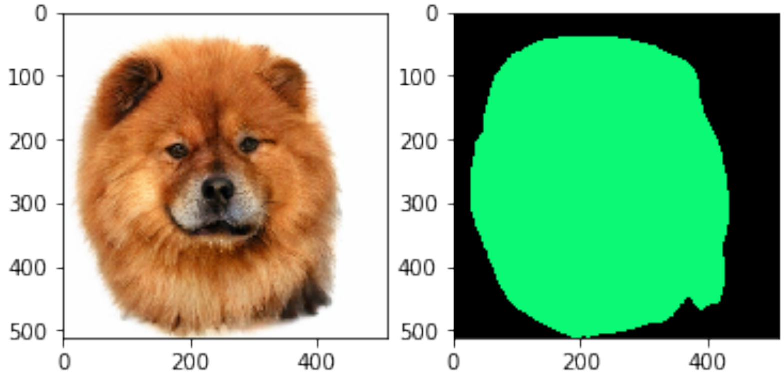
```
cost 7  is  tensor(0.6068, grad_fn=<NllLoss2DBackward>)
```



```
cost 8  is  tensor(0.5387, grad_fn=<NllLoss2DBackward>)
```



```
cost 9  is  tensor(0.5050, grad_fn=<NllLoss2DBackward>)
```



In [13]:

```
# # 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")

# # plot the semantic segmentation predictions of 21 classes in each color
# r = Image.fromarray(output_predictions.byte().cpu().numpy()).resize(input_image.size)
# r.putpalette(colors)

# import matplotlib.pyplot as plt
# plt.imshow(r)
# # plt.show()
```