how_to_use_checkpoint

November 23, 2025

0.1 How to use a checkpoint

[19]:
```python
from unet import UNet
import torch
if torch.cuda.is_available():
    device = torch.device('cuda')
elif torch.backends.mps.is_available():
    device = torch.device('mps') # for mac
else:
    device = torch.device('cpu')
print(device)
net = UNet(n_channels=3, n_classes=16, bilinear=False)
net.to(device=device) #       GPU

checkpoint_path = './checkpoints/checkpoint_epoch1.pth'
state_dict = torch.load(checkpoint_path, map_location=device)  #  torch.load

#    keys
print(state_dict.keys())
print(len(state_dict.keys()))
```

```
mps
odict_keys(['inc.double_conv.0.weight', 'inc.double_conv.1.weight',
'inc.double_conv.1.bias', 'inc.double_conv.1.running_mean',
'inc.double_conv.1.running_var', 'inc.double_conv.1.num_batches_tracked',
'inc.double_conv.3.weight', 'inc.double_conv.4.weight',
'inc.double_conv.4.bias', 'inc.double_conv.4.running_mean',
'inc.double_conv.4.running_var', 'inc.double_conv.4.num_batches_tracked',
'down1.maxpool_conv.1.double_conv.0.weight',
'down1.maxpool_conv.1.double_conv.1.weight',
'down1.maxpool_conv.1.double_conv.1.bias',
'down1.maxpool_conv.1.double_conv.1.running_mean',
'down1.maxpool_conv.1.double_conv.1.running_var',
'down1.maxpool_conv.1.double_conv.1.num_batches_tracked',
'down1.maxpool_conv.1.double_conv.3.weight',
'down1.maxpool_conv.1.double_conv.4.weight',
'down1.maxpool_conv.1.double_conv.4.bias',
'down1.maxpool_conv.1.double_conv.4.running_mean',
'down1.maxpool_conv.1.double_conv.4.running_var',
'down1.maxpool_conv.1.double_conv.4.num_batches_tracked',
'down2.maxpool_conv.1.double_conv.0.weight',
'down2.maxpool_conv.1.double_conv.1.weight',
'down2.maxpool_conv.1.double_conv.1.bias',
'down2.maxpool_conv.1.double_conv.1.running_mean',
'down2.maxpool_conv.1.double_conv.1.running_var',
'down2.maxpool_conv.1.double_conv.1.num_batches_tracked',
'down2.maxpool_conv.1.double_conv.3.weight',
'down2.maxpool_conv.1.double_conv.4.weight',
'down2.maxpool_conv.1.double_conv.4.bias',
'down2.maxpool_conv.1.double_conv.4.running_mean',
'down2.maxpool_conv.1.double_conv.4.running_var',
'down2.maxpool_conv.1.double_conv.4.num_batches_tracked',
'down3.maxpool_conv.1.double_conv.0.weight',
'down3.maxpool_conv.1.double_conv.1.weight',
'down3.maxpool_conv.1.double_conv.1.bias',
'down3.maxpool_conv.1.double_conv.1.running_mean',
'down3.maxpool_conv.1.double_conv.1.running_var',
'down3.maxpool_conv.1.double_conv.1.num_batches_tracked',
'down3.maxpool_conv.1.double_conv.3.weight',
'down3.maxpool_conv.1.double_conv.4.weight',
'down3.maxpool_conv.1.double_conv.4.bias',
'down3.maxpool_conv.1.double_conv.4.running_mean',
'down3.maxpool_conv.1.double_conv.4.running_var',
'down3.maxpool_conv.1.double_conv.4.num_batches_tracked',
'down4.maxpool_conv.1.double_conv.0.weight',
'down4.maxpool_conv.1.double_conv.1.weight',
'down4.maxpool_conv.1.double_conv.1.bias',
'down4.maxpool_conv.1.double_conv.1.running_mean',
'down4.maxpool_conv.1.double_conv.1.running_var',
'down4.maxpool_conv.1.double_conv.1.num_batches_tracked',
'down4.maxpool_conv.1.double_conv.3.weight',
'down4.maxpool_conv.1.double_conv.4.weight',
'down4.maxpool_conv.1.double_conv.4.bias',
'down4.maxpool_conv.1.double_conv.4.running_mean',
'down4.maxpool_conv.1.double_conv.4.running_var',
'down4.maxpool_conv.1.double_conv.4.num_batches_tracked', 'up1.up.weight',
'up1.up.bias', 'up1.conv.double_conv.0.weight', 'up1.conv.double_conv.1.weight',
'up1.conv.double_conv.1.bias', 'up1.conv.double_conv.1.running_mean',
'up1.conv.double_conv.1.running_var',
'up1.conv.double_conv.1.num_batches_tracked', 'up1.conv.double_conv.3.weight',
'up1.conv.double_conv.4.weight', 'up1.conv.double_conv.4.bias',
'up1.conv.double_conv.4.running_mean', 'up1.conv.double_conv.4.running_var',
'up1.conv.double_conv.4.num_batches_tracked', 'up2.up.weight', 'up2.up.bias',
'up2.conv.double_conv.0.weight', 'up2.conv.double_conv.1.weight',
'up2.conv.double_conv.1.bias', 'up2.conv.double_conv.1.running_mean',
'up2.conv.double_conv.1.running_var',
'up2.conv.double_conv.1.num_batches_tracked', 'up2.conv.double_conv.3.weight',
'up2.conv.double_conv.4.weight', 'up2.conv.double_conv.4.bias',
'up2.conv.double_conv.4.running_mean', 'up2.conv.double_conv.4.running_var',
'up2.conv.double_conv.4.num_batches_tracked', 'up3.up.weight', 'up3.up.bias',
'up3.conv.double_conv.0.weight', 'up3.conv.double_conv.1.weight',
'up3.conv.double_conv.1.bias', 'up3.conv.double_conv.1.running_mean',
'up3.conv.double_conv.1.running_var',
'up3.conv.double_conv.1.num_batches_tracked', 'up3.conv.double_conv.3.weight',
'up3.conv.double_conv.4.weight', 'up3.conv.double_conv.4.bias',
'up3.conv.double_conv.4.running_mean', 'up3.conv.double_conv.4.running_var',
'up3.conv.double_conv.4.num_batches_tracked', 'up4.up.weight', 'up4.up.bias',
'up4.conv.double_conv.0.weight', 'up4.conv.double_conv.1.weight',
'up4.conv.double_conv.1.bias', 'up4.conv.double_conv.1.running_mean',
'up4.conv.double_conv.1.running_var',
'up4.conv.double_conv.1.num_batches_tracked', 'up4.conv.double_conv.3.weight',
'up4.conv.double_conv.4.weight', 'up4.conv.double_conv.4.bias',
'up4.conv.double_conv.4.running_mean', 'up4.conv.double_conv.4.running_var',
'up4.conv.double_conv.4.num_batches_tracked', 'outc.conv.weight',
'outc.conv.bias', 'mask_values'])
119
```

[20]:
```
# status_dict        'mask_values',
state_dict.pop('mask_values', [0, 1])
print(state_dict.keys())
print(len(state_dict.keys()))
```

```
odict_keys(['inc.double_conv.0.weight', 'inc.double_conv.1.weight',
'inc.double_conv.1.bias', 'inc.double_conv.1.running_mean',
'inc.double_conv.1.running_var', 'inc.double_conv.1.num_batches_tracked',
'inc.double_conv.3.weight', 'inc.double_conv.4.weight',
'inc.double_conv.4.bias', 'inc.double_conv.4.running_mean',
'inc.double_conv.4.running_var', 'inc.double_conv.4.num_batches_tracked',
'down1.maxpool_conv.1.double_conv.0.weight',
'down1.maxpool_conv.1.double_conv.1.weight',
'down1.maxpool_conv.1.double_conv.1.bias',
'down1.maxpool_conv.1.double_conv.1.running_mean',
'down1.maxpool_conv.1.double_conv.1.running_var',
'down1.maxpool_conv.1.double_conv.1.num_batches_tracked',
'down1.maxpool_conv.1.double_conv.3.weight',
'down1.maxpool_conv.1.double_conv.4.weight',
'down1.maxpool_conv.1.double_conv.4.bias',
'down1.maxpool_conv.1.double_conv.4.running_mean',
'down1.maxpool_conv.1.double_conv.4.running_var',
'down1.maxpool_conv.1.double_conv.4.num_batches_tracked',
'down2.maxpool_conv.1.double_conv.0.weight',
'down2.maxpool_conv.1.double_conv.1.weight',
'down2.maxpool_conv.1.double_conv.1.bias',
'down2.maxpool_conv.1.double_conv.1.running_mean',
'down2.maxpool_conv.1.double_conv.1.running_var',
'down2.maxpool_conv.1.double_conv.1.num_batches_tracked',
'down2.maxpool_conv.1.double_conv.3.weight',
'down2.maxpool_conv.1.double_conv.4.weight',
'down2.maxpool_conv.1.double_conv.4.bias',
'down2.maxpool_conv.1.double_conv.4.running_mean',
'down2.maxpool_conv.1.double_conv.4.running_var',
'down2.maxpool_conv.1.double_conv.4.num_batches_tracked',
'down3.maxpool_conv.1.double_conv.0.weight',
'down3.maxpool_conv.1.double_conv.1.weight',
'down3.maxpool_conv.1.double_conv.1.bias',
'down3.maxpool_conv.1.double_conv.1.running_mean',
'down3.maxpool_conv.1.double_conv.1.running_var',
'down3.maxpool_conv.1.double_conv.1.num_batches_tracked',
'down3.maxpool_conv.1.double_conv.3.weight',
'down3.maxpool_conv.1.double_conv.4.weight',
'down3.maxpool_conv.1.double_conv.4.bias',
'down3.maxpool_conv.1.double_conv.4.running_mean',
'down3.maxpool_conv.1.double_conv.4.running_var',
'down3.maxpool_conv.1.double_conv.4.num_batches_tracked',
'down4.maxpool_conv.1.double_conv.0.weight',
'down4.maxpool_conv.1.double_conv.1.weight',
'down4.maxpool_conv.1.double_conv.1.bias',
'down4.maxpool_conv.1.double_conv.1.running_mean',
'down4.maxpool_conv.1.double_conv.1.running_var',
'down4.maxpool_conv.1.double_conv.1.num_batches_tracked',
'down4.maxpool_conv.1.double_conv.3.weight',
'down4.maxpool_conv.1.double_conv.4.weight',
'down4.maxpool_conv.1.double_conv.4.bias',
'down4.maxpool_conv.1.double_conv.4.running_mean',
'down4.maxpool_conv.1.double_conv.4.running_var',
'down4.maxpool_conv.1.double_conv.4.num_batches_tracked', 'up1.up.weight',
'up1.up.bias', 'up1.conv.double_conv.0.weight', 'up1.conv.double_conv.1.weight',
'up1.conv.double_conv.1.bias', 'up1.conv.double_conv.1.running_mean',
'up1.conv.double_conv.1.running_var',
'up1.conv.double_conv.1.num_batches_tracked', 'up1.conv.double_conv.3.weight',
'up1.conv.double_conv.4.weight', 'up1.conv.double_conv.4.bias',
'up1.conv.double_conv.4.running_mean', 'up1.conv.double_conv.4.running_var',
'up1.conv.double_conv.4.num_batches_tracked', 'up2.up.weight', 'up2.up.bias',
'up2.conv.double_conv.0.weight', 'up2.conv.double_conv.1.weight',
'up2.conv.double_conv.1.bias', 'up2.conv.double_conv.1.running_mean',
'up2.conv.double_conv.1.running_var',
'up2.conv.double_conv.1.num_batches_tracked', 'up2.conv.double_conv.3.weight',
'up2.conv.double_conv.4.weight', 'up2.conv.double_conv.4.bias',
'up2.conv.double_conv.4.running_mean', 'up2.conv.double_conv.4.running_var',
'up2.conv.double_conv.4.num_batches_tracked', 'up3.up.weight', 'up3.up.bias',
'up3.conv.double_conv.0.weight', 'up3.conv.double_conv.1.weight',
'up3.conv.double_conv.1.bias', 'up3.conv.double_conv.1.running_mean',
'up3.conv.double_conv.1.running_var',
'up3.conv.double_conv.1.num_batches_tracked', 'up3.conv.double_conv.3.weight',
'up3.conv.double_conv.4.weight', 'up3.conv.double_conv.4.bias',
'up3.conv.double_conv.4.running_mean', 'up3.conv.double_conv.4.running_var',
'up3.conv.double_conv.4.num_batches_tracked', 'up4.up.weight', 'up4.up.bias',
'up4.conv.double_conv.0.weight', 'up4.conv.double_conv.1.weight',
'up4.conv.double_conv.1.bias', 'up4.conv.double_conv.1.running_mean',
'up4.conv.double_conv.1.running_var',
'up4.conv.double_conv.1.num_batches_tracked', 'up4.conv.double_conv.3.weight',
'up4.conv.double_conv.4.weight', 'up4.conv.double_conv.4.bias',
'up4.conv.double_conv.4.running_mean', 'up4.conv.double_conv.4.running_var',
'up4.conv.double_conv.4.num_batches_tracked', 'outc.conv.weight',
'outc.conv.bias'])
118
```

[21]:

```
from utils.data_loading import WSDataset
from unet import UNet
from utils.utils import plot_img_and_mask
import torch
import torch.nn.functional as F
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
from pathlib import Path
from torchvision import transforms

# data_loading
class ImageResize(torch.nn.Module):
    def __init__(self, new_size, interpolate_mode=Image.NEAREST):
        super(ImageResize, self).__init__()
        self.new_size = new_size
        self.interpolate_mode = interpolate_mode # Image.NEAREST if is_mask␣
 ↪else Image.BICUBIC

    def forward(self, img):
        # img = transforms.Resize(self.new_size, interpolation=self.
 ↪interpolate_mode)
        img = img.resize(self.new_size, resample=self.interpolate_mode)
        img = np.asarray(img)
        return img


class ImageNormalization(torch.nn.Module):
    def __init__(self):
        super(ImageNormalization, self).__init__()

    def forward(self, img):
        img = img / 255.0
        return img
```

[22]:
```
net.load_state_dict(state_dict)
#
def predict_img(net,
                full_img,
                device,
                scale_factor=1,
                out_threshold=0.5):
    net.eval()
    new_size = (full_img.size[0] * scale_factor, full_img.size[1] *␣
 ↪scale_factor)
    im_transform = transforms.Compose([ #
        ImageResize(new_size, interpolate_mode=Image.BICUBIC),
        ImageNormalization(),
        transforms.ToTensor()
    ])
    img = im_transform(full_img).unsqueeze(0)
    img = img.to(device=device, dtype=torch.float32)
    print(img.shape)

    with torch.no_grad():
        output = net(img).cpu()
        output = F.interpolate(output, (full_img.size[1], full_img.size[0]),␣
 ↪mode='bilinear')
        mask = output.argmax(dim=1)
    return mask[0].long().squeeze().numpy()
```