```
x  # real images
G  # generator
D  # discriminator

# Minimize logits for generated images
logits = D(G(x))
loss_d_adv_fake = F.softplus(logits)

# Maximize logits for real images
logits = D(x)
loss_d_adv_real = F.softplus(-logits)

# MS3D regularization
grads = torch.autograd.grad(outputs=logits, inputs=x)
grads = normalize(grads)
grads = reshape_to_square(grads)
spatial_size = get_spatial_size(grads)
ms3d = 0
for i in range(torch.log2(spatial_size)):
    grads_down = F.avg_pool2d(grads, kernel_size=2, stride=2)  # Kadanoff decimation
    # or use: grads_down = F.conv2d(grads, guass_kernel, stride=2)
    grads_recn = F.interpolate(grads_down, size=grads.shape[2:], mode='nearest')
    ms3d += F.mse_loss(grads, grads_recn)
    grads = grads_down
loss_d = loss_d_adv_fake + loss_d_adv_real + ms3d
loss_d.backward()
```