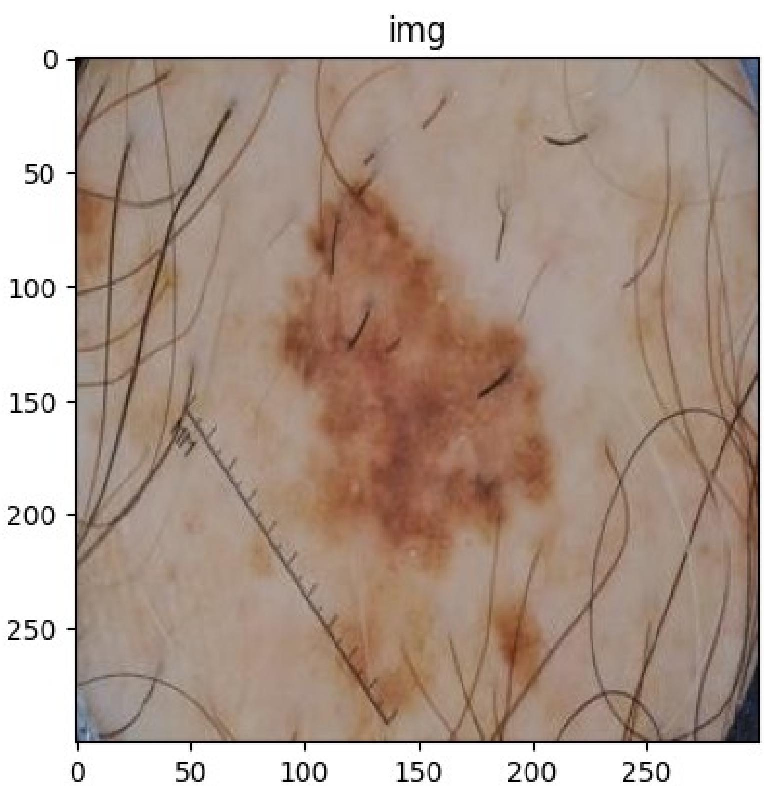
```
import torch
from torchvision import models, transforms
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
import numpy as np
from torch.utils.data import Dataset, DataLoader
from PIL import Image
from torchvision.transforms import ToPILImage
from torchvision.datasets import ImageFolder
import torch.nn.functional as F
import random

import os
os.environ['KAGGLE_CONFIG_DIR']='/content'

#https://www.kaggle.com/datasets/hasnainjaved/melanoma-skin-cancer-dataset-of-10000-images
!kaggle datasets download -d hasnainjaved/melanoma-skin-cancer-dataset-of-10000-images

Dataset URL: https://www.kaggle.com/datasets/hasnainjaved/melanoma-skin-cancer-dataset-of-10000-images
License(s): CC0-1.0
Downloading melanoma-skin-cancer-dataset-of-10000-images.zip to /content
 99% 98.0M/98.7M [00:01<00:00, 83.7MB/s]
100% 98.7M/98.7M [00:01<00:00, 77.4MB/s]

!unzip -q \*.zip && rm *.zip

data_set_train = []
data_set_test = []

train_beign = '/content/melanoma_cancer_dataset/train/benign'
test_beign = '/content/melanoma_cancer_dataset/test/benign'
train_malignant = '/content/melanoma_cancer_dataset/train/malignant'
test_malignant = '/content/melanoma_cancer_dataset/test/malignant'

contents1 = os.listdir(train_beign)
contents2 = os.listdir(test_beign)
contents3 = os.listdir(train_malignant)
contents4 = os.listdir(test_malignant)

for item in contents1:
  data_set_train.append((Image.open(os.path.join(train_beign, item)), 0))

for item in contents2:
  data_set_test.append((Image.open(os.path.join(test_beign, item)), 0))
```

```python
for item in contents3:
  data_set_train.append((Image.open(os.path.join(train_malignant, item)), 1))

for item in contents4:
  data_set_test.append((Image.open(os.path.join(test_malignant, item)), 1))

random.shuffle(data_set_train)
random.shuffle(data_set_test)

print(len(data_set_train), len(data_set_test))
```

```
9605 1000
```

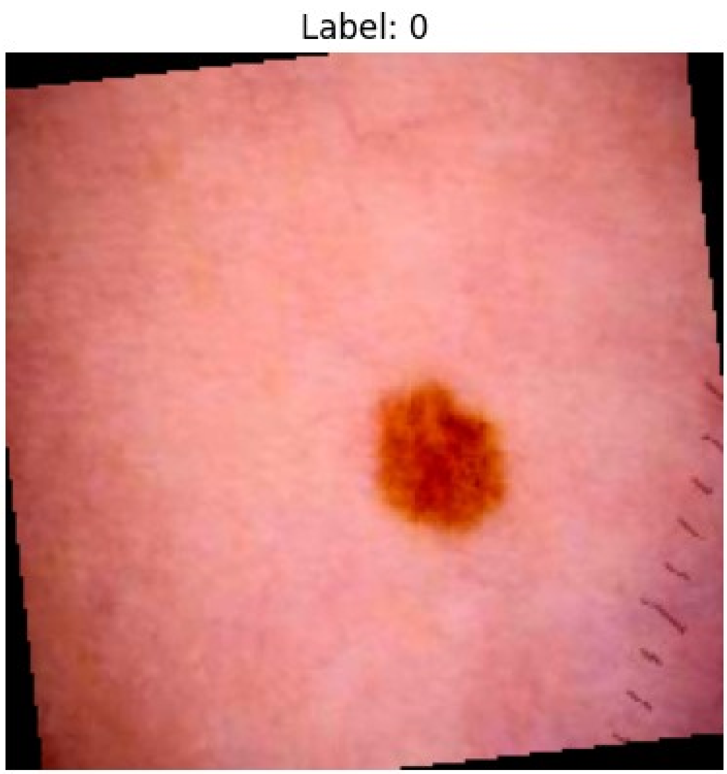
```python
# Assuming data_set_train is a list of (image, label) tuples
train_img, train_label = random.choice(data_set_train)

fig = plt.figure(figsize=(10, 10))
ax = plt.subplot(1, 2, 1)
ax.imshow(train_img)
ax.set_title('img')
plt.show()
```



```
class SkinLesions(Dataset):
    def __init__(self, mode, data_set_train, data_set_test):
        self.mode = mode

        if self.mode == 'train':
            self.images = [item[0] for item in data_set_train]
            self.labels = [item[1] for item in data_set_train]
        elif self.mode == 'test':
            self.images = [item[0] for item in data_set_test]
            self.labels = [item[1] for item in data_set_test]
        else:
            raise ValueError('Invalid mode')

        self.transform = transforms.Compose([
            transforms.Resize((200, 200)),
            transforms.RandomHorizontalFlip(),
            transforms.RandomVerticalFlip(),
            transforms.RandomRotation(degrees=10),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
        ])

    def __len__(self):
        return len(self.images)

    def __getitem__(self, idx):
        image = self.images[idx]
        if isinstance(image, str):
            image = Image.open(image).convert('RGB')
        image = self.transform(image)
        label = torch.tensor(self.labels[idx], dtype=torch.long)  #
Ensure label is a tensor
        return image, label

batch_size = 20
learning_rate = 0.001
epochs = 10

train_set = SkinLesions('train', data_set_train, data_set_test)
test_set = SkinLesions('test', data_set_train, data_set_test)

train_dataloader = DataLoader(train_set, batch_size=batch_size,
shuffle=True)
test_dataloader = DataLoader(test_set, batch_size=batch_size,
shuffle=False)

# Iterate over the train_dataloader
counter = 0
for idx, (images, labels) in enumerate(train_dataloader):
    # Access the first batch
```

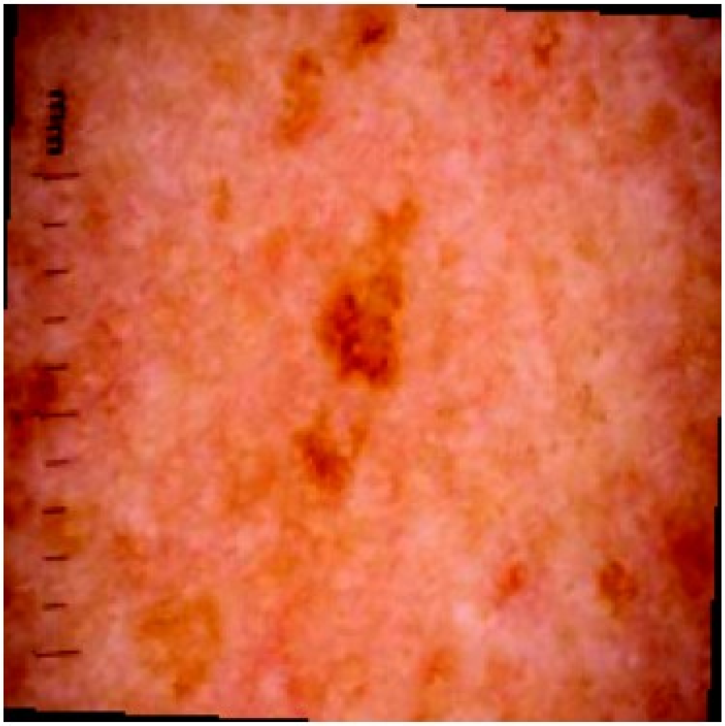
```
    # Get the first image and label from the batch
    image = images[idx]
    label = labels[idx]


    # Convert the image tensor to a NumPy array
    image_np = image.permute(1, 2, 0).cpu().numpy()

    # Display the image using Matplotlib
    plt.imshow(image_np)
    plt.title(f'Label: {label.item()}')
    plt.axis('off')
    plt.show()
    if counter == 3:
      break
    else:
      counter = counter +1
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for
imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



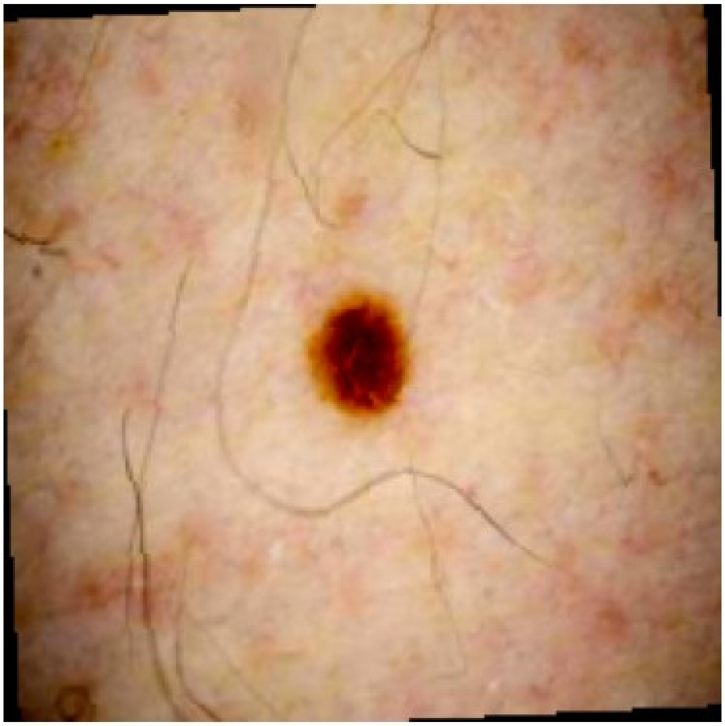

```
WARNING:matplotlib.image:Clipping input data to the valid range for
imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Label: 0




```
WARNING:matplotlib.image:Clipping input data to the valid range for
imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Label: 0



```
WARNING:matplotlib.image:Clipping input data to the valid range for
imshow with RGB data ([0..1] for floats or [0..255] for integers).
```


```
class SkinLesions(nn.Module):
    def __init__(self):
        super(SkinLesions, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=32,
kernel_size=3, stride=1, padding=1)
        self.bn1 = nn.BatchNorm2d(32)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)

        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64,
kernel_size=3, stride=1, padding=1)
        self.bn2 = nn.BatchNorm2d(64)

        self.fc1 = nn.Linear(160000, 128)
        self.fc2 = nn.Linear(128, 2)

    def forward(self, x):
        x = self.pool(torch.relu(self.bn1(self.conv1(x))))
        x = self.pool(torch.relu(self.bn2(self.conv2(x))))
        flatten = nn.Flatten()
        x = flatten(x)
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x


model = SkinLesions().to('cuda')
```

```
model = model.to('cuda')
criterion = nn.CrossEntropyLoss()
optim = torch.optim.SGD(model.parameters(), lr=learning_rate,
momentum=0.9)

train_loss_per_epoch = []
for epoch in range(10):
    running_loss_train = []
    model.train()
    total_correct = 0
    total_samples = 0

    for inputs, labels in train_dataloader:
        inputs = inputs.to('cuda')
        labels = labels.to('cuda')

        optim.zero_grad()

        outputs = model(inputs)
        loss = criterion(outputs, labels)

        loss.backward()
        optim.step()

        running_loss_train.append(loss.item())
        _, predicted = torch.max(outputs, 1)
        total_correct += (predicted == labels).sum().item()
        total_samples += labels.size(0)
    avg_loss = np.mean(running_loss_train)
    accuracy = 100 * total_correct / total_samples
    train_loss_per_epoch.append(avg_loss)

    print(f'Epoch [{epoch + 1}/10], Batch Losses: {avg_loss},
Accuracy: {accuracy}')

print('Finished Training')
```

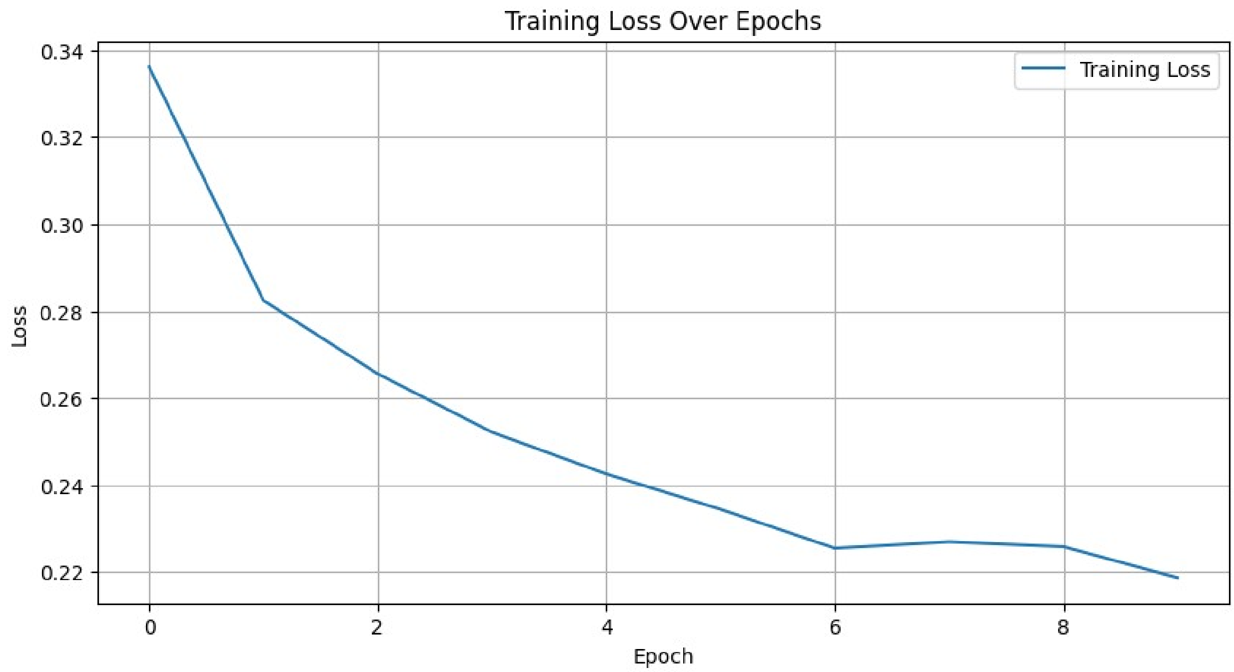
```
Epoch [1/10], Batch Losses: 0.33616194601428234, Accuracy:
85.66371681415929
Epoch [2/10], Batch Losses: 0.2824573282717791, Accuracy:
88.07912545549193
Epoch [3/10], Batch Losses: 0.2655923709722542, Accuracy:
88.70380010411245
Epoch [4/10], Batch Losses: 0.25216612482362133, Accuracy:
89.2243623112962
Epoch [5/10], Batch Losses: 0.2425702603421563, Accuracy:
89.69286829776158
Epoch [6/10], Batch Losses: 0.23444230158633228, Accuracy:
90.29672045809474
Epoch [7/10], Batch Losses: 0.22548546363249142, Accuracy:
```

```
90.76522644456013
Epoch [8/10], Batch Losses: 0.22695786663158768, Accuracy:
90.29672045809474
Epoch [9/10], Batch Losses: 0.22584870220655712, Accuracy:
90.58823529411765
Epoch [10/10], Batch Losses: 0.2187057258154647, Accuracy:
90.87975013014055
Finished Training
```

```
with torch.no_grad():
    model.eval()
    correct = 0
    total = 0
    running_loss_test = []
    for inputs, labels in test_dataloader:
        inputs = inputs.to('cuda')
        labels = labels.to('cuda')

        outputs = model(inputs)
        loss = criterion(outputs, labels)
        running_loss_test.append(loss.item())


        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    avg_loss = np.mean(running_loss_test)

    print(f'Test Loss: {avg_loss}, Accuracy: {100 * correct / total}')
```

```
Test Loss: 0.228204225897789, Accuracy: 90.9
```

```
epochs = range(len(train_loss_per_epoch))

# Plotting training loss
plt.figure(figsize=(10, 5))
plt.plot(epochs, train_loss_per_epoch, label='Training Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Over Epochs')
plt.legend()
plt.grid(True)
plt.show()
```




```python
# Print model's state_dict
print("Model's state_dict:")
for param_tensor in model.state_dict():
    print(param_tensor, "\t", model.state_dict()[param_tensor].size())

# Print optimizer's state_dict
print("Optimizer's state_dict:")
for var_name in optim.state_dict():
    print(var_name, "\t", optim.state_dict()[var_name])
```

```
Model's state_dict:
conv1.weight     torch.Size([32, 3, 3, 3])
conv1.bias       torch.Size([32])
bn1.weight       torch.Size([32])
bn1.bias    torch.Size([32])
bn1.running_mean       torch.Size([32])
bn1.running_var  torch.Size([32])
bn1.num_batches_tracked     torch.Size([])
conv2.weight     torch.Size([64, 32, 3, 3])
conv2.bias       torch.Size([64])
bn2.weight       torch.Size([64])
bn2.bias    torch.Size([64])
bn2.running_mean       torch.Size([64])
bn2.running_var  torch.Size([64])
bn2.num_batches_tracked     torch.Size([])
fc1.weight       torch.Size([128, 160000])
fc1.bias    torch.Size([128])
fc2.weight       torch.Size([2, 128])
fc2.bias    torch.Size([2])
```

```
Optimizer's state_dict:
state 	 {0: {'momentum_buffer': tensor([[[[ 1.0399e-01,  3.4982e-
02,  4.6634e-02],
          [ 9.9428e-02,  1.8068e-02,  3.1556e-02],
          [ 9.6770e-02,  1.7678e-02,  3.0029e-02]],

         [[ 1.8940e-01,  1.6640e-01,  1.8730e-01],
          [ 1.7974e-01,  1.4639e-01,  1.6814e-01],
          [ 1.7361e-01,  1.4190e-01,  1.6197e-01]],

         [[ 2.0293e-01,  2.0444e-01,  2.2990e-01],
          [ 1.9163e-01,  1.8351e-01,  2.0991e-01],
          [ 1.8336e-01,  1.7871e-01,  2.0250e-01]]],


        [[[-5.7630e-02, -4.1755e-02, -9.0863e-02],
          [-3.2206e-02, -1.4184e-02, -5.7375e-02],
          [-5.0927e-02, -2.2261e-02, -5.8852e-02]],

         [[-1.2982e-01, -1.1811e-01, -1.5503e-01],
          [-1.1628e-01, -1.0272e-01, -1.3471e-01],
          [-1.2854e-01, -1.0577e-01, -1.3076e-01]],

         [[-7.0530e-03,  5.1351e-03, -2.4685e-02],
          [ 1.6851e-03,  1.7301e-02, -7.7426e-03],
          [-5.8906e-03,  1.7458e-02, -6.9400e-04]]],


        [[[ 1.8781e-01,  1.8687e-01,  1.6769e-01],
          [ 1.7524e-01,  1.7132e-01,  1.6239e-01],
          [ 1.7171e-01,  1.6896e-01,  1.6155e-01]],

         [[ 1.0759e-01,  1.0715e-01,  8.4279e-02],
          [ 8.5450e-02,  8.4081e-02,  6.8369e-02],
          [ 7.8717e-02,  8.0674e-02,  6.2743e-02]],

         [[ 8.7628e-02,  8.9764e-02,  6.9249e-02],
          [ 6.1510e-02,  6.5140e-02,  5.1362e-02],
          [ 5.4279e-02,  6.1113e-02,  4.2022e-02]]],


        [[[ 9.3216e-02,  4.4748e-02,  5.0367e-02],
          [ 8.1067e-02,  2.3899e-02,  5.1910e-02],
          [ 8.4001e-02,  3.3820e-02,  5.5866e-02]],

         [[ 1.9599e-01,  1.7004e-01,  1.7606e-01],
          [ 1.8113e-01,  1.4696e-01,  1.6920e-01],
          [ 1.7997e-01,  1.5089e-01,  1.6781e-01]],

         [[ 2.3133e-01,  2.2366e-01,  2.3082e-01],
          [ 2.1567e-01,  1.9917e-01,  2.1818e-01],
```

```
          [ 2.1596e-01,  2.0021e-01,  2.1454e-01]]],


        [[[-5.7307e-02,  9.2303e-02,  2.4431e-03],
          [-9.9157e-02,  3.9191e-02, -3.0501e-02],
          [-2.7575e-01, -1.4011e-01, -2.1348e-01]],

         [[-6.6875e-02,  4.4726e-02, -2.5930e-02],
          [-1.0628e-01, -3.5509e-03, -5.7667e-02],
          [-2.1599e-01, -1.1086e-01, -1.7461e-01]],

         [[-8.1435e-02,  9.2438e-03, -5.3639e-02],
          [-1.1967e-01, -3.6341e-02, -8.4393e-02],
          [-1.9112e-01, -1.0131e-01, -1.6069e-01]]],


        [[[-7.9498e-03, -1.5776e-02, -5.4431e-03],
          [-5.0438e-03, -1.6753e-02, -5.7698e-03],
          [ 1.1969e-02,  1.5279e-03,  7.9696e-03]],

         [[-1.5958e-02, -2.1118e-02, -1.6547e-02],
          [-1.6003e-02, -2.3227e-02, -1.8227e-02],
          [-3.4474e-03, -9.8636e-03, -7.8349e-03]],

         [[ 6.3402e-03,  2.5432e-03,  3.4504e-03],
          [ 4.6984e-03, -9.4494e-05,  1.0087e-03],
          [ 1.5954e-02,  1.1625e-02,  1.0298e-02]]],


        [[[ 1.5024e-01,  6.2309e-02,  6.7632e-02],
          [ 1.3652e-01,  4.9654e-02,  6.3000e-02],
          [ 1.1396e-01,  3.9616e-02,  4.6044e-02]],

         [[ 2.0307e-01,  1.6210e-01,  1.6959e-01],
          [ 1.8839e-01,  1.4768e-01,  1.6279e-01],
          [ 1.6830e-01,  1.3560e-01,  1.4476e-01]],

         [[ 2.1855e-01,  2.0492e-01,  2.1484e-01],
          [ 2.0533e-01,  1.9204e-01,  2.0948e-01],
          [ 1.8848e-01,  1.8146e-01,  1.9349e-01]]],


        [[[-3.7855e-02, -3.7410e-02, -4.3346e-02],
          [-3.3485e-02, -4.0441e-02, -4.4210e-02],
          [-1.4095e-02, -1.0304e-02, -1.1201e-02]],

         [[-6.6657e-02, -6.4976e-02, -6.9903e-02],
          [-6.4507e-02, -6.8056e-02, -7.1181e-02],
          [-5.1112e-02, -4.6264e-02, -4.6436e-02]],

         [[-2.9521e-02, -2.6976e-02, -3.1461e-02],
```

```
          [-2.8362e-02, -3.0417e-02, -3.3317e-02],
          [-1.7576e-02, -1.2215e-02, -1.2495e-02]]],


        [[[-5.2431e-03, -6.9485e-03,  8.1095e-02],
          [-1.3296e-02, -1.1863e-02,  8.5008e-02],
          [-5.9260e-02, -5.2098e-02,  5.3945e-02]],

         [[-3.9386e-02, -3.6389e-02,  1.4955e-02],
          [-4.8062e-02, -4.0992e-02,  1.4443e-02],
          [-7.9188e-02, -6.7314e-02, -7.2434e-03]],

         [[ 2.5524e-02,  2.8631e-02,  6.0521e-02],
          [ 1.4443e-02,  2.2241e-02,  5.4941e-02],
          [-7.6958e-03,  4.8438e-03,  4.0428e-02]]],


        [[[ 3.7573e-02,  2.9778e-02,  3.1394e-02],
          [ 3.4580e-02,  1.7653e-02,  2.3416e-02],
          [ 3.3029e-02,  2.8313e-02,  3.0612e-02]],

         [[ 1.3955e-01,  1.3254e-01,  1.3579e-01],
          [ 1.3702e-01,  1.2327e-01,  1.3048e-01],
          [ 1.3703e-01,  1.3201e-01,  1.3680e-01]],

         [[ 1.7229e-01,  1.6538e-01,  1.6981e-01],
          [ 1.7047e-01,  1.5768e-01,  1.6593e-01],
          [ 1.7295e-01,  1.6658e-01,  1.7250e-01]]],


        [[[-2.0809e-02, -2.3966e-02, -2.8917e-02],
          [-1.0160e-02, -1.2184e-02, -1.4704e-02],
          [-1.4415e-02, -1.4222e-02, -1.4649e-02]],

         [[ 1.4484e-02,  1.2363e-02,  8.1192e-03],
          [ 2.0576e-02,  1.9312e-02,  1.7257e-02],
          [ 1.6021e-02,  1.6287e-02,  1.5952e-02]],

         [[ 7.0180e-02,  6.7699e-02,  6.2897e-02],
          [ 7.4266e-02,  7.2971e-02,  7.0569e-02],
          [ 6.9703e-02,  7.0069e-02,  6.9882e-02]]],


        [[[-9.1935e-02, -9.1217e-02, -1.0991e-01],
          [-1.0152e-01, -1.1430e-01, -1.4052e-01],
          [-9.6650e-02, -1.0584e-01, -1.1746e-01]],

         [[-1.3706e-01, -1.4323e-01, -1.5120e-01],
          [-1.4599e-01, -1.6151e-01, -1.7436e-01],
          [-1.4434e-01, -1.5740e-01, -1.5741e-01]],
```

```
          [[-1.4025e-01, -1.5147e-01, -1.5682e-01],
           [-1.4912e-01, -1.6691e-01, -1.7592e-01],
           [-1.4856e-01, -1.6531e-01, -1.6211e-01]]],


         [[[ 7.6565e-03, -3.2169e-02, -2.9538e-02],
           [ 2.2154e-02, -1.1019e-02, -1.0474e-02],
           [ 2.4196e-02, -2.5699e-03, -1.7133e-04]],

          [[ 5.3280e-02,  2.6666e-02,  2.9535e-02],
           [ 6.5339e-02,  4.2760e-02,  4.4684e-02],
           [ 6.7027e-02,  4.9703e-02,  5.1563e-02]],

          [[ 1.4015e-01,  1.2202e-01,  1.2511e-01],
           [ 1.5110e-01,  1.3727e-01,  1.4070e-01],
           [ 1.5230e-01,  1.4411e-01,  1.4793e-01]]],


         [[[ 1.5984e-01,  1.6090e-01,  1.6411e-01],
           [ 1.5352e-01,  1.6216e-01,  1.7106e-01],
           [ 1.3691e-01,  1.4647e-01,  1.5461e-01]],

          [[-1.2577e-02, -6.7626e-03, -7.6812e-03],
           [-1.4629e-02, -2.3803e-03, -4.8374e-04],
           [-3.7024e-02, -2.5687e-02, -2.4981e-02]],

          [[-1.0462e-01, -9.9883e-02, -1.0107e-01],
           [-1.0744e-01, -9.5726e-02, -9.5057e-02],
           [-1.3379e-01, -1.2199e-01, -1.2376e-01]]],


         [[[-1.9452e-03, -1.4697e-02, -7.4495e-03],
           [-5.0967e-03, -1.6302e-02, -7.4264e-03],
           [-2.0359e-02, -3.4093e-02, -2.4565e-02]],

          [[ 1.0079e-02,  2.5231e-03,  8.8769e-03],
           [ 6.1483e-03, -1.4602e-04,  7.8887e-03],
           [-6.2531e-03, -1.4217e-02, -6.2032e-03]],

          [[ 1.1398e-02,  6.8994e-03,  1.2711e-02],
           [ 7.9793e-03,  4.6015e-03,  1.1716e-02],
           [-1.5130e-03, -6.3183e-03,  2.2341e-04]]],


         [[[-2.4685e-02, -2.8846e-02, -4.7636e-02],
           [-1.9115e-02, -2.3886e-02, -4.2280e-02],
           [-6.5847e-04, -5.2546e-03, -2.2907e-02]],

          [[ 2.2534e-02,  2.0649e-02,  1.2897e-02],
           [ 2.4976e-02,  2.3465e-02,  1.6722e-02],
           [ 3.8155e-02,  3.6716e-02,  3.0300e-02]],
```

```
         [[ 3.8338e-02,  3.7021e-02,  3.3586e-02],
          [ 3.9217e-02,  3.8437e-02,  3.6464e-02],
          [ 4.9673e-02,  4.8755e-02,  4.6532e-02]]],


        [[[ 2.5368e-03,  1.4037e-02,  1.4269e-02],
          [-5.4637e-03,  3.4006e-03,  2.5545e-03],
          [-2.1695e-02, -5.8830e-03, -6.9632e-03]],

         [[ 9.4674e-02,  9.7099e-02,  1.0014e-01],
          [ 8.5905e-02,  8.7172e-02,  8.8803e-02],
          [ 7.5084e-02,  8.1450e-02,  8.3749e-02]],

         [[ 1.3396e-01,  1.3096e-01,  1.3551e-01],
          [ 1.2372e-01,  1.2034e-01,  1.2351e-01],
          [ 1.1459e-01,  1.1595e-01,  1.2013e-01]]],


        [[[-3.1730e-02, -2.7925e-02, -2.5585e-02],
          [-4.0185e-02, -3.7536e-02, -3.4715e-02],
          [-2.9127e-02, -2.8038e-02, -2.2833e-02]],

         [[ 1.5331e-02,  1.7475e-02,  1.9203e-02],
          [ 6.9170e-03,  8.3095e-03,  1.0899e-02],
          [ 1.3652e-02,  1.3981e-02,  1.8528e-02]],

         [[ 2.7117e-02,  2.7934e-02,  2.9284e-02],
          [ 1.8635e-02,  1.8852e-02,  2.0916e-02],
          [ 2.3798e-02,  2.3063e-02,  2.7491e-02]]],


        [[[ 2.2207e-01,  1.6084e-01,  2.9015e-01],
          [ 1.9105e-01,  1.5365e-01,  3.0446e-01],
          [ 1.0940e-01,  8.7901e-02,  2.2642e-01]],

         [[ 3.2013e-01,  2.8537e-01,  3.5316e-01],
          [ 2.8449e-01,  2.6994e-01,  3.5590e-01],
          [ 2.1867e-01,  2.1160e-01,  2.9511e-01]],

         [[ 3.6558e-01,  3.4876e-01,  3.8468e-01],
          [ 3.2779e-01,  3.3013e-01,  3.8291e-01],
          [ 2.7194e-01,  2.7551e-01,  3.3074e-01]]],


        [[[-3.8122e-02, -4.3418e-02, -4.3016e-02],
          [-4.5708e-02, -4.9166e-02, -4.9695e-02],
          [-5.2339e-02, -5.2110e-02, -5.4104e-02]],

         [[-4.0533e-02, -4.3650e-02, -3.9295e-02],
          [-4.6600e-02, -4.7997e-02, -4.4155e-02],
```

```
          [-5.3297e-02, -5.1080e-02, -4.8814e-02]],

         [[ 2.1647e-02,  1.9931e-02,  2.5643e-02],
          [ 1.6986e-02,  1.6780e-02,  2.1922e-02],
          [ 1.0918e-02,  1.4636e-02,  1.7740e-02]]],


        [[[ 2.9026e-01,  2.7302e-01,  3.0512e-01],
          [ 3.0005e-01,  2.7867e-01,  3.1814e-01],
          [ 3.0033e-01,  2.8705e-01,  3.2788e-01]],

         [[ 1.9407e-01,  1.8348e-01,  2.0441e-01],
          [ 1.9899e-01,  1.8711e-01,  2.1375e-01],
          [ 1.9613e-01,  1.8903e-01,  2.1811e-01]],

         [[ 1.7197e-01,  1.6563e-01,  1.8088e-01],
          [ 1.7460e-01,  1.6966e-01,  1.9074e-01],
          [ 1.7209e-01,  1.6988e-01,  1.9425e-01]]],


        [[[ 1.8035e-01,  1.9425e-01,  1.2846e-01],
          [ 2.8375e-01,  3.0368e-01,  2.2793e-01],
          [ 1.5094e-01,  1.7590e-01,  1.2023e-01]],

         [[ 8.0205e-02,  9.0704e-02,  2.8851e-02],
          [ 1.7641e-01,  1.9465e-01,  1.2492e-01],
          [ 6.7554e-02,  9.0629e-02,  3.6292e-02]],

         [[-3.8866e-02, -2.9260e-02, -8.7416e-02],
          [ 4.8654e-02,  6.6115e-02,  8.8557e-04],
          [-5.0404e-02, -2.8552e-02, -8.0374e-02]]],


        [[[ 8.0641e-02,  1.1098e-01,  2.6910e-01],
          [ 2.5509e-02,  6.1403e-02,  2.1969e-01],
          [-1.3848e-01, -1.0194e-01,  6.7312e-02]],

         [[ 1.4068e-01,  1.5622e-01,  2.1794e-01],
          [ 9.3114e-02,  1.1294e-01,  1.7230e-01],
          [-2.4826e-02, -6.9713e-03,  5.9933e-02]],

         [[ 2.2916e-01,  2.3595e-01,  2.5384e-01],
          [ 1.8635e-01,  1.9792e-01,  2.1547e-01],
          [ 9.1029e-02,  1.0013e-01,  1.2341e-01]]],


        [[[-3.5607e-02, -4.2149e-02, -3.3054e-02],
          [ 4.8623e-03, -2.2849e-04,  6.2552e-03],
          [-9.5438e-03, -7.7652e-03, -3.5172e-03]],

         [[-2.5309e-02, -2.9973e-02, -2.7535e-02],
```

```
          [-1.2465e-03, -3.7492e-03, -3.0403e-03],
          [-1.4816e-02, -1.1409e-02, -1.3017e-02]],

         [[-2.7573e-02, -3.2373e-02, -3.4375e-02],
          [-9.3233e-03, -1.3101e-02, -1.6358e-02],
          [-2.1256e-02, -1.9419e-02, -2.5505e-02]]],


        [[[ 1.9752e-02,  1.3684e-02,  6.5163e-02],
          [ 1.4983e-02,  1.8344e-02,  7.5778e-02],
          [ 1.7491e-02,  2.7613e-02,  8.6999e-02]],

         [[ 3.8793e-02,  3.1125e-02,  5.2404e-02],
          [ 3.1644e-02,  3.1611e-02,  5.7524e-02],
          [ 2.9617e-02,  3.6090e-02,  6.2837e-02]],

         [[ 2.5432e-02,  1.6988e-02,  2.2464e-02],
          [ 1.7769e-02,  1.6524e-02,  2.6319e-02],
          [ 1.6305e-02,  2.1167e-02,  3.1628e-02]]],


        [[[ 1.4945e-02,  1.8100e-02,  2.0237e-02],
          [ 1.8284e-02,  3.7686e-02,  2.6112e-02],
          [ 1.4781e-02,  2.9006e-02,  1.7354e-02]],

         [[ 6.0168e-02,  6.5274e-02,  6.4650e-02],
          [ 6.1010e-02,  7.6895e-02,  6.5474e-02],
          [ 5.4060e-02,  6.5509e-02,  5.3557e-02]],

         [[ 6.2056e-02,  6.6995e-02,  6.3802e-02],
          [ 6.2896e-02,  7.6792e-02,  6.4123e-02],
          [ 5.4391e-02,  6.5342e-02,  5.3567e-02]]],


        [[[ 7.6983e-02,  7.4155e-02,  4.9738e-02],
          [ 6.8175e-02,  6.1328e-02,  3.7300e-02],
          [ 1.4234e-01,  1.2897e-01,  1.0914e-01]],

         [[ 2.4948e-02,  2.6205e-02,  7.0482e-03],
          [ 2.1188e-02,  1.8238e-02, -7.8898e-04],
          [ 7.5105e-02,  6.7132e-02,  5.1083e-02]],

         [[ 4.5240e-03,  8.0809e-03, -9.4257e-03],
          [ 3.2994e-04,  3.2475e-04, -1.6621e-02],
          [ 4.1259e-02,  3.5935e-02,  2.1459e-02]]],


        [[[ 5.5777e-02,  7.1830e-02,  7.3625e-02],
          [ 1.7791e-02,  3.7100e-02,  4.2840e-02],
          [-3.2303e-02, -2.8672e-02, -1.8846e-02]],
```

```
          [[ 7.3014e-02,  9.2925e-02,  9.8487e-02],
           [ 3.9242e-02,  6.1587e-02,  6.8661e-02],
           [ 8.7237e-03,  1.8358e-02,  2.8573e-02]],

          [[-4.2907e-02, -2.2897e-02, -1.8157e-02],
           [-7.2419e-02, -5.0431e-02, -4.5604e-02],
           [-9.2846e-02, -8.1433e-02, -7.2862e-02]]],


         [[[-1.1054e-02, -2.1725e-02, -2.6031e-02],
           [-3.2801e-02, -3.9334e-02, -4.2556e-02],
           [-3.4943e-02, -4.7044e-02, -5.2447e-02]],

          [[ 2.2384e-02,  1.5290e-02,  1.3462e-02],
           [ 5.3467e-03,  1.6259e-03,  8.1042e-04],
           [ 4.4174e-03, -3.3067e-03, -5.9674e-03]],

          [[ 3.0046e-02,  2.4887e-02,  2.4041e-02],
           [ 1.5566e-02,  1.2915e-02,  1.2808e-02],
           [ 1.6157e-02,  9.8716e-03,  7.9519e-03]]],


         [[[-5.9823e-02, -6.6095e-02, -4.7854e-02],
           [-4.1050e-02, -4.9086e-02, -3.1843e-02],
           [-1.8809e-02, -2.7681e-02, -1.0115e-02]],

          [[-2.6129e-02, -2.9617e-02, -1.2837e-02],
           [-1.3914e-02, -1.9590e-02, -2.5227e-03],
           [ 8.4417e-04, -6.2841e-03,  1.1190e-02]],

          [[ 4.9583e-02,  4.7036e-02,  6.3538e-02],
           [ 5.9034e-02,  5.3505e-02,  7.0517e-02],
           [ 7.1796e-02,  6.4169e-02,  8.1914e-02]]],


         [[[ 7.7414e-02,  9.1867e-02,  1.0598e-01],
           [ 7.7823e-02,  1.0103e-01,  1.1299e-01],
           [ 3.4472e-02,  5.3220e-02,  6.6445e-02]],

          [[ 3.2754e-02,  4.2147e-02,  5.2365e-02],
           [ 4.1736e-02,  5.7853e-02,  6.4907e-02],
           [ 1.7384e-02,  2.9223e-02,  3.7742e-02]],

          [[ 1.7130e-02,  2.2210e-02,  2.9435e-02],
           [ 2.9237e-02,  4.0352e-02,  4.5327e-02],
           [ 1.4110e-02,  2.1773e-02,  3.0544e-02]]],


         [[[ 1.5345e-03, -4.2292e-03, -4.4901e-03],
           [-8.8985e-03, -1.3077e-02, -1.3078e-02],
           [-2.4132e-02, -3.0211e-02, -3.0537e-02]],
```

```
         [[ 2.1446e-02,  1.7086e-02,  1.7381e-02],
          [ 1.4092e-02,  1.1008e-02,  1.1340e-02],
          [ 5.0357e-03,  1.2554e-04,  5.3408e-04]],

         [[ 2.8776e-02,  2.4870e-02,  2.5892e-02],
          [ 2.2737e-02,  2.0184e-02,  2.1069e-02],
          [ 1.6319e-02,  1.1804e-02,  1.3122e-02]]]],
device='cuda:0')}, 1: {'momentum_buffer': tensor([-8.0449e-08, -
1.1061e-08,  3.9129e-08,  6.3175e-08, -3.5843e-08,
         9.4916e-09, -5.0082e-09, -1.3343e-08,  7.9279e-08,  3.5606e-
10,
         2.7682e-09, -3.2021e-08,  2.7027e-08, -2.2066e-08,  6.1306e-
10,
        -1.2541e-08,  1.7665e-09,  6.6312e-09,  4.9504e-09,  3.8286e-
09,
         6.2356e-08, -1.3496e-07, -8.3970e-08,  2.2490e-08,  1.5992e-
08,
         1.7803e-08, -6.5544e-09, -3.5174e-08,  3.1769e-09, -2.5251e-
08,
         4.2418e-08, -1.5312e-08], device='cuda:0')}, 2:
{'momentum_buffer': tensor([ 0.0366,  0.0781,  0.0092,  0.0107,
0.0605, -0.0307, -0.0347,  0.0417,
         0.0620, -0.0033,  0.0065,  0.0375, -0.0348,  0.0310, -0.0515,
-0.0007,
        -0.0166, -0.0256,  0.0125,  0.0249,  0.0368, -0.0576, -0.1748,
0.0332,
        -0.0013, -0.0149,  0.0536,  0.0292, -0.0347, -0.0661,  0.0303,
-0.0436],
       device='cuda:0')}, 3: {'momentum_buffer': tensor([-0.0313,
0.0447,  0.0284, -0.0404, -0.0215, -0.0008, -0.0755,  0.0084,
         0.0291, -0.0510, -0.0146,  0.0111, -0.1143,  0.0769, -0.0086,
0.0055,
        -0.0411, -0.0162, -0.0330,  0.0160,  0.0651, -0.0262, -0.0935,
0.0214,
        -0.0553, -0.0604,  0.0447,  0.0258, -0.0202, -0.0157,  0.0068,
-0.0267],
       device='cuda:0')}, 4: {'momentum_buffer': tensor([[[[ 0.0112,
0.0112,  0.0080],
          [ 0.0108,  0.0118,  0.0084],
          [ 0.0117,  0.0131,  0.0093]],

         [[ 0.0042,  0.0061,  0.0044],
          [ 0.0043,  0.0042,  0.0031],
          [ 0.0087,  0.0100,  0.0083]],

         [[-0.0228, -0.0283, -0.0339],
          [-0.0343, -0.0353, -0.0402],
          [-0.0275, -0.0370, -0.0430]],
```

```
         ...,

         [[-0.0318, -0.0355, -0.0425],
          [-0.0328, -0.0351, -0.0406],
          [-0.0266, -0.0309, -0.0378]],

         [[-0.0149, -0.0159, -0.0227],
          [-0.0244, -0.0252, -0.0327],
          [-0.0184, -0.0230, -0.0297]],

         [[-0.0303, -0.0356, -0.0398],
          [-0.0328, -0.0390, -0.0431],
          [-0.0303, -0.0384, -0.0427]]],


        [[[ 0.0351,  0.0338,  0.0339],
          [ 0.0293,  0.0330,  0.0306],
          [ 0.0195,  0.0236,  0.0222]],

         [[ 0.0432,  0.0405,  0.0401],
          [ 0.0310,  0.0324,  0.0380],
          [ 0.0365,  0.0349,  0.0387]],

         [[ 0.0017, -0.0068, -0.0069],
          [ 0.0330,  0.0372,  0.0307],
          [ 0.0071,  0.0100,  0.0061]],

         ...,

         [[-0.0073, -0.0107, -0.0118],
          [ 0.0199,  0.0182,  0.0142],
          [ 0.0143,  0.0126,  0.0065]],

         [[ 0.0497,  0.0460,  0.0325],
          [ 0.0833,  0.0823,  0.0785],
          [ 0.0618,  0.0547,  0.0532]],

         [[ 0.0261,  0.0199,  0.0117],
          [ 0.0481,  0.0439,  0.0391],
          [ 0.0453,  0.0399,  0.0359]]],


        [[[-0.0054, -0.0040, -0.0068],
          [-0.0053, -0.0033, -0.0061],
          [-0.0041, -0.0024, -0.0055]],

         [[ 0.0021,  0.0060,  0.0132],
          [-0.0017,  0.0042,  0.0111],
          [ 0.0026,  0.0076,  0.0150]],

         [[-0.0136, -0.0063, -0.0073],
```

```
          [-0.0207, -0.0143, -0.0135],
          [-0.0231, -0.0162, -0.0167]],

         ...,

         [[-0.0203, -0.0138, -0.0183],
          [-0.0197, -0.0127, -0.0183],
          [-0.0226, -0.0149, -0.0201]],

         [[ 0.0003,  0.0101,  0.0071],
          [-0.0044,  0.0048,  0.0007],
          [-0.0072,  0.0017, -0.0036]],

         [[-0.0139, -0.0093, -0.0109],
          [-0.0136, -0.0090, -0.0102],
          [-0.0169, -0.0125, -0.0153]]],


        ...,


        [[[-0.0197, -0.0230, -0.0251],
          [-0.0147, -0.0198, -0.0210],
          [-0.0157, -0.0225, -0.0229]],

         [[ 0.0809,  0.0810,  0.0804],
          [ 0.0946,  0.0929,  0.0906],
          [ 0.0822,  0.0804,  0.0804]],

         [[ 0.0330,  0.0301,  0.0292],
          [ 0.0372,  0.0356,  0.0350],
          [ 0.0271,  0.0247,  0.0255]],

         ...,

         [[ 0.0354,  0.0309,  0.0324],
          [ 0.0416,  0.0370,  0.0381],
          [ 0.0314,  0.0267,  0.0303]],

         [[ 0.0227,  0.0151,  0.0211],
          [ 0.0285,  0.0231,  0.0283],
          [ 0.0093,  0.0090,  0.0146]],

         [[ 0.0311,  0.0275,  0.0296],
          [ 0.0333,  0.0309,  0.0328],
          [ 0.0241,  0.0237,  0.0258]]],


        [[[ 0.0147,  0.0161,  0.0152],
          [ 0.0154,  0.0158,  0.0160],
          [ 0.0136,  0.0140,  0.0158]],
```

```
          [[ 0.0085,  0.0141,  0.0170],
           [ 0.0180,  0.0208,  0.0262],
           [ 0.0075,  0.0113,  0.0147]],

          [[-0.0234, -0.0107, -0.0103],
           [-0.0160, -0.0090, -0.0033],
           [-0.0201, -0.0138, -0.0086]],

          ...,

          [[-0.0381, -0.0297, -0.0262],
           [-0.0357, -0.0274, -0.0247],
           [-0.0354, -0.0271, -0.0253]],

          [[-0.0326, -0.0257, -0.0223],
           [-0.0210, -0.0174, -0.0137],
           [-0.0309, -0.0291, -0.0285]],

          [[-0.0445, -0.0360, -0.0345],
           [-0.0396, -0.0326, -0.0314],
           [-0.0415, -0.0355, -0.0357]]],


         [[[-0.0104, -0.0117, -0.0141],
           [-0.0107, -0.0119, -0.0137],
           [-0.0116, -0.0144, -0.0180]],

          [[ 0.0750,  0.0811,  0.0988],
           [ 0.0836,  0.0900,  0.1039],
           [ 0.0687,  0.0765,  0.0935]],

          [[-0.1152, -0.1176, -0.1045],
           [-0.1143, -0.1109, -0.0973],
           [-0.1228, -0.1179, -0.1078]],

          ...,

          [[-0.1389, -0.1399, -0.1355],
           [-0.1361, -0.1369, -0.1326],
           [-0.1335, -0.1322, -0.1273]],

          [[-0.0496, -0.0436, -0.0384],
           [-0.0302, -0.0196, -0.0175],
           [-0.0498, -0.0387, -0.0361]],

          [[-0.1152, -0.1127, -0.1086],
           [-0.1050, -0.0991, -0.0958],
           [-0.1120, -0.1067, -0.1022]]]], device='cuda:0')}, 5:
{'momentum_buffer': tensor([-4.4348e-10,  1.2843e-08,  1.5627e-09, -
2.2448e-09, -9.8176e-11,
```

```
         1.4826e-09,  1.6664e-09, -1.1192e-09,  6.7759e-09,  1.1521e-
08,
         1.2243e-09, -1.8554e-09, -8.8903e-11,  3.1683e-09,  9.4719e-
09,
        -3.1067e-09,  9.6906e-09,  1.3537e-09, -9.1013e-09,  1.7897e-
09,
        -3.5888e-09, -6.1096e-09,  3.8859e-09,  3.1439e-09,  4.3054e-
11,
         2.7116e-10,  7.3244e-09,  7.0231e-09, -6.6523e-09, -1.6583e-
08,
         7.9609e-09, -3.4172e-09, -5.8400e-09,  7.0621e-09, -1.8794e-
09,
         5.4489e-09,  3.0018e-09, -7.1387e-09, -2.1493e-09,  4.3530e-
09,
        -1.3895e-09, -7.4441e-09, -3.2068e-09, -2.5831e-09,  7.5821e-
10,
         8.2290e-09, -5.8410e-09,  2.3614e-09, -7.5229e-09, -4.4823e-
09,
        -2.2979e-09,  3.1590e-09, -5.8237e-09, -2.7887e-09, -2.1682e-
09,
        -5.0083e-09, -3.9344e-09, -8.2355e-09, -1.0950e-08,  2.2519e-
10,
        -4.8350e-10, -8.9297e-09, -1.7761e-09, -3.7669e-09],
device='cuda:0')}, 6: {'momentum_buffer': tensor([ 0.0289,  0.0550, -
0.0279,  0.0018, -0.0135, -0.0256, -0.0631, -0.0561,
         0.0468,  0.0193,  0.0033,  0.0370,  0.0259, -0.0118,  0.0183,
0.0538,
         0.0656, -0.0018,  0.0048,  0.0278,  0.0072,  0.0253,  0.0024,
0.0177,
         0.0168, -0.0080,  0.0338,  0.0191, -0.0073,  0.0354,  0.0798,
0.0037,
        -0.0301,  0.0325, -0.0461,  0.0288, -0.0531, -0.0378,  0.0611,
0.0462,
        -0.0250, -0.0446, -0.0679,  0.0517, -0.0099,  0.0793, -0.0053,
0.0210,
        -0.0053, -0.0196,  0.0679,  0.0534, -0.0036, -0.0071,  0.0118,
-0.0202,
        -0.0070, -0.0106, -0.0091,  0.0144,  0.0106, -0.0398,  0.0146,
-0.0593],
       device='cuda:0')}, 7: {'momentum_buffer': tensor([ 0.0160,
0.0443, -0.0128,  0.0061, -0.0109, -0.0484, -0.0402, -0.0134,
         0.0413,  0.0111, -0.0015,  0.0116,  0.0123, -0.0274, -0.0162,
0.0175,
         0.0536,  0.0015, -0.0200,  0.0146,  0.0034,  0.0034,  0.0041,
-0.0083,
         0.0223, -0.0018,  0.0216,  0.0167, -0.0373,  0.0172,  0.0590,
0.0103,
        -0.0268,  0.0194, -0.0535,  0.0297, -0.0524, -0.0421,  0.0294,
0.0348,
```

```
        -0.0591, -0.1154, -0.0662,  0.0200,  0.0052,  0.0889,  0.0068,
-0.0066,
         0.0200, -0.0382,  0.0367,  0.0303, -0.0146, -0.0313,  0.0050,
-0.0304,
         0.0024, -0.0243, -0.0220,  0.0200, -0.0088, -0.0867,  0.0161,
-0.0261],
       device='cuda:0')}, 8: {'momentum_buffer': tensor([[ 5.6052e-45,
5.6052e-45,  5.6052e-45,  ...,  5.6052e-45,
          5.6052e-45,  5.6052e-45],
        [ 4.5940e-05,  3.7694e-05,  5.4337e-06,  ...,  1.3392e-06,
          1.3118e-06,  1.0450e-05],
        [ 5.6052e-45,  5.6052e-45,  5.6052e-45,  ...,  5.6052e-45,
          5.6052e-45,  5.6052e-45],
        ...,
        [ 5.6052e-45,  5.6052e-45,  5.6052e-45,  ...,  5.6052e-45,
          5.6052e-45,  5.6052e-45],
        [-7.7325e-04, -7.5033e-04, -6.8785e-04,  ..., -2.6016e-04,
         -2.0409e-04, -3.3676e-04],
        [ 5.6052e-45,  5.6052e-45,  5.6052e-45,  ...,  5.6052e-45,
          5.6052e-45,  5.6052e-45]], device='cuda:0')}, 9:
{'momentum_buffer': tensor([ 5.6052e-45,  1.0514e-05,  5.6052e-45,
5.6052e-45, -1.1824e-05,
        -8.2854e-04,  5.6052e-45,  5.6052e-45,  3.1972e-05,  3.6466e-
17,
        -1.7109e-09, -5.6052e-45,  4.9660e-03,  5.6052e-45, -5.6052e-
45,
         6.8408e-05, -4.8619e-06,  3.5626e-03,  5.6052e-45, -1.0961e-
02,
         5.6052e-45,  5.2291e-04,  5.6052e-45,  5.6052e-45,  5.6052e-
45,
        -1.6462e-03, -3.2860e-04,  5.6052e-45,  5.6052e-45,  5.6052e-
45,
         5.6052e-45,  4.0660e-03,  5.6052e-45,  5.6052e-45,  5.6052e-
45,
        -2.2755e-13,  6.1682e-04, -6.3858e-05,  3.0368e-05,  5.6052e-
45,
         3.9138e-03,  5.6052e-45,  5.6052e-45, -3.7595e-03,  5.6052e-
45,
         5.6052e-45,  5.6052e-45, -3.6337e-03, -5.6052e-45,  5.6052e-
45,
        -1.4409e-13,  1.1422e-06,  5.8803e-03,  1.7915e-03,  5.6052e-
45,
         5.6159e-04,  5.6052e-45,  5.6052e-45,  9.8488e-05,  5.6052e-
45,
         5.6052e-45, -3.0370e-04,  5.6052e-45,  5.6052e-45, -5.6052e-
45,
         1.0406e-03,  5.0457e-13,  5.6052e-45,  5.1738e-06,  1.3901e-
06,
        -1.6713e-04,  5.6052e-45,  6.2297e-06, -4.3281e-05, -4.8865e-
```

```
03,
        2.0269e-04,  5.6052e-45,  5.6052e-45,  1.2744e-20,  5.6052e-
45,
       -1.3191e-08,  5.6052e-45,  5.6052e-45, -1.5488e-03,  2.1257e-
03,
        1.1407e-03,  5.6052e-45, -8.4567e-07, -9.1460e-05, -2.4834e-
04,
       -7.0530e-05, -4.0875e-07,  5.6052e-45,  5.6052e-45, -4.9491e-
07,
        5.6052e-45,  5.6052e-45, -1.0610e-02,  5.6052e-45,  5.6052e-
45,
       -6.8320e-03, -7.2305e-04,  1.4022e-04,  2.0617e-02,  1.9994e-
02,
        5.6052e-45,  5.6052e-45, -2.4412e-32,  5.6052e-45,  5.6052e-
45,
        6.3688e-03,  5.6052e-45, -1.1253e-05,  5.6052e-45,  5.6052e-
45,
        5.6052e-45, -5.6052e-45,  5.6052e-45,  7.2041e-04, -2.1124e-
06,
        5.6052e-45, -2.7306e-06,  5.6052e-45, -8.0978e-04,  1.9310e-
03,
        5.6052e-45, -3.7932e-04,  5.6052e-45], device='cuda:0')}, 10:
{'momentum_buffer': tensor([[ 5.6052e-45, -5.0596e-04, -5.6052e-45,
5.6052e-45, -1.5907e-04,
         6.9726e-02,  5.6052e-45, -5.6052e-45,  4.6524e-04,  8.6567e-
17,
         1.2695e-09, -5.6052e-45,  2.3218e-01,  5.6052e-45, -5.6052e-
45,
        -3.5905e-01, -1.4853e-04,  1.5520e-01, -5.6052e-45,  6.6960e-
01,
         5.6052e-45, -2.0632e-02,  5.6052e-45, -5.6052e-45,  5.6052e-
45,
         1.9208e-01,  1.8167e-02, -5.6052e-45, -5.6052e-45, -5.6052e-
45,
         5.6052e-45, -9.7856e-02, -5.6052e-45, -5.6052e-45,  5.6052e-
45,
         1.0680e-13,  9.6358e-02, -1.1193e-03,  3.4983e-03,  5.6052e-
45,
         3.8966e-01, -5.6052e-45,  5.6052e-45,  2.3339e-01,  5.6052e-
45,
        -5.6052e-45,  5.6052e-45,  1.9848e-01, -5.6052e-45,  5.6052e-
45,
        -1.4167e-13,  3.3327e-02,  7.1003e-01,  1.7246e-01, -5.6052e-
45,
        -4.6739e-03, -5.6052e-45,  5.6052e-45,  1.7046e-02,  5.6052e-
45,
         5.6052e-45,  8.8367e-02,  5.6052e-45, -5.6052e-45, -5.6052e-
45,
         2.9362e-01,  3.1190e-12,  5.6052e-45, -1.0270e-05,  8.4499e-
```

```
03,
         -1.8256e-02, -5.6052e-45, -1.7811e-06,  5.3888e-04, -1.6716e-
01,
         -1.7118e-02,  5.6052e-45, -5.6052e-45, -1.2147e-20, -5.6052e-
45,
          1.8999e-07, -5.6052e-45, -5.6052e-45,  1.9279e-01,  1.5871e-
01,
         -4.9282e-02,  5.6052e-45, -6.8213e-07, -3.4069e-03,  5.9331e-
03,
          3.1041e-03,  1.1494e-06, -5.6052e-45,  5.6052e-45,  7.4825e-
06,
         -5.6052e-45, -5.6052e-45,  6.9596e-01,  5.6052e-45, -5.6052e-
45,
          8.4628e-01,  1.4482e-02,  6.2264e-03,  2.6550e+00,
1.9323e+00,
          5.6052e-45, -5.6052e-45,  4.7521e-32,  5.6052e-45, -5.6052e-
45,
          2.2088e-01, -5.6052e-45,  2.0261e-03,  5.6052e-45, -5.6052e-
45,
          5.6052e-45,  5.6052e-45, -5.6052e-45,  3.6617e-01,  1.4008e-
05,
          5.6052e-45, -5.6016e-06,  5.6052e-45, -2.6568e-01, -7.5424e-
02,
         -5.6052e-45, -1.3018e-02, -5.6052e-45],
        [-5.6052e-45,  5.0596e-04,  5.6052e-45, -5.6052e-45,  1.5907e-
04,
         -6.9726e-02, -5.6052e-45,  5.6052e-45, -4.6522e-04, -8.6557e-
17,
         -1.4776e-09,  5.6052e-45, -2.3218e-01, -5.6052e-45,  5.6052e-
45,
          3.5905e-01,  1.4853e-04, -1.5520e-01,  5.6052e-45, -6.6960e-
01,
         -5.6052e-45,  2.0632e-02, -5.6052e-45,  5.6052e-45, -5.6052e-
45,
         -1.9208e-01, -1.8167e-02,  5.6052e-45,  5.6052e-45,  5.6052e-
45,
         -5.6052e-45,  9.7856e-02,  5.6052e-45,  5.6052e-45, -5.6052e-
45,
         -1.0642e-13, -9.6358e-02,  1.1193e-03, -3.4983e-03, -5.6052e-
45,
         -3.8965e-01,  5.6052e-45, -5.6052e-45, -2.3339e-01, -5.6052e-
45,
          5.6052e-45, -5.6052e-45, -1.9848e-01,  5.6052e-45, -5.6052e-
45,
          1.4196e-13, -3.3327e-02, -7.1003e-01, -1.7246e-01,  5.6052e-
45,
          4.6739e-03,  5.6052e-45, -5.6052e-45, -1.7046e-02, -5.6052e-
45,
         -5.6052e-45, -8.8367e-02, -5.6052e-45,  5.6052e-45,  5.6052e-
```

```
45,
        -2.9362e-01, -3.1190e-12, -5.6052e-45,  1.0270e-05, -8.4499e-
03,
         1.8256e-02,  5.6052e-45,  1.7811e-06, -5.3888e-04,  1.6716e-
01,
         1.7118e-02, -5.6052e-45,  5.6052e-45,  1.2147e-20,  5.6052e-
45,
        -1.8970e-07,  5.6052e-45,  5.6052e-45, -1.9279e-01, -1.5871e-
01,
         4.9282e-02, -5.6052e-45,  6.8211e-07,  3.4069e-03, -5.9329e-
03,
        -3.1040e-03, -1.1494e-06,  5.6052e-45, -5.6052e-45, -7.4825e-
06,
         5.6052e-45,  5.6052e-45, -6.9596e-01, -5.6052e-45,  5.6052e-
45,
        -8.4628e-01, -1.4482e-02, -6.2264e-03, -2.6550e+00, -
1.9323e+00,
        -5.6052e-45,  5.6052e-45, -4.6769e-32, -5.6052e-45,  5.6052e-
45,
        -2.2088e-01,  5.6052e-45, -2.0261e-03, -5.6052e-45,  5.6052e-
45,
        -5.6052e-45, -5.6052e-45,  5.6052e-45, -3.6617e-01, -1.4007e-
05,
        -5.6052e-45,  5.6016e-06, -5.6052e-45,  2.6568e-01,  7.5424e-
02,
         5.6052e-45,  1.3018e-02,  5.6052e-45]], device='cuda:0')},
11: {'momentum_buffer': tensor([ 0.2795, -0.2795], device='cuda:0')}}
param_groups     [{'lr': 0.001, 'momentum': 0.9, 'dampening': 0,
'weight_decay': 0, 'nesterov': False, 'maximize': False, 'foreach':
None, 'differentiable': False, 'fused': None, 'params': [0, 1, 2, 3,
4, 5, 6, 7, 8, 9, 10, 11]}]
```

```
from google.colab import drive
drive.mount('/content/drive')
```

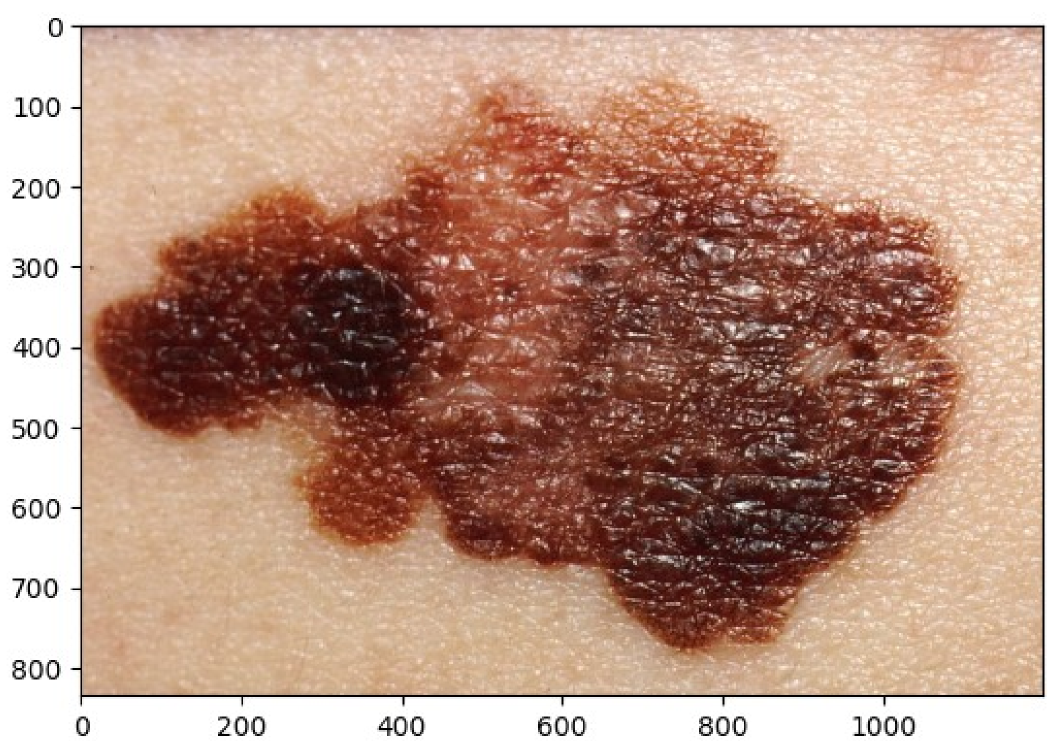
```
Mounted at /content/drive
```

```
#dont run this
torch.save(model.state_dict(),
'/content/drive/MyDrive/model_weights.pth')
print("Model weights saved successfully!")
```

```
Model weights saved successfully!
```

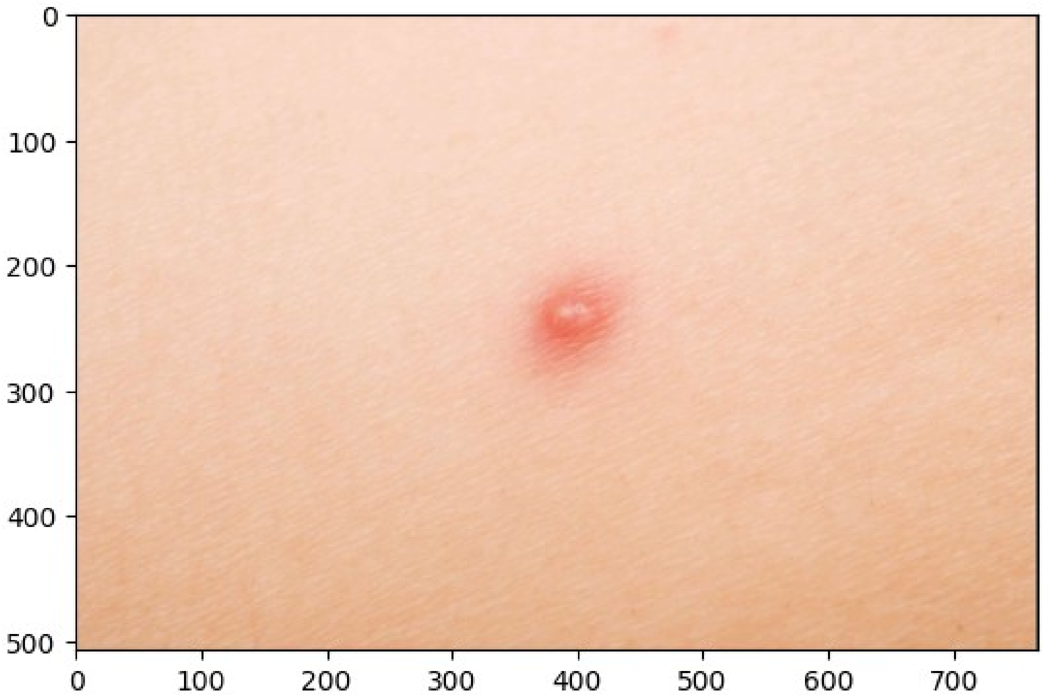
```
import matplotlib.pyplot as plt

#Testing image from web - Melanoma
model = SkinLesions()
model.load_state_dict(torch.load('/content/drive/MyDrive/Skin_lesions_
project/model_weights.pth'))
```

```
model.eval()

image_path =
'/content/drive/MyDrive/Skin_lesions_project/Melanoma.jpg' #Sample
image from the web (Melanoma)
image = Image.open(image_path).convert('RGB')
plt.imshow(image)
plt.show()
transform = transforms.Compose([
            transforms.Resize((200, 200)),
            transforms.RandomHorizontalFlip(),
            transforms.RandomVerticalFlip(),
            transforms.RandomRotation(degrees=10),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
        ])
image = transform(image)
with torch.no_grad():
    output = model(image.unsqueeze(0))
    _, predicted = torch.max(output, 1)
    if predicted.item() == 1:
      print('Melanoma')
    else:
      print('Benign')
```




```
Melanoma
```

```
#Testing image from web - No Melanoma
model = SkinLesions()
model.load_state_dict(torch.load('/content/drive/MyDrive/Skin_lesions_
project/model_weights.pth'))
model.eval()

image_path = '/content/drive/MyDrive/Skin_lesions_project/pimple.png'
#Sample image from the web (acnae on the skin)
image = Image.open(image_path).convert('RGB')
plt.imshow(image)
plt.show()
transform = transforms.Compose([
            transforms.Resize((200, 200)),
            transforms.RandomHorizontalFlip(),
            transforms.RandomVerticalFlip(),
            transforms.RandomRotation(degrees=10),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
        ])
image = transform(image)

with torch.no_grad():
    output = model(image.unsqueeze(0))
    _, predicted = torch.max(output, 1)
    if predicted.item() == 1:
      print('Melanoma')
    else:
      print('Benign')
```




```
Benign
```