Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

По дисциплине «ОИвИС»

Тема: “Обучение классификаторов средствами библиотеки PyTorch”

Выполнил:

Студент 4 курса

Группы ИИ-23

Скварнюк Д.Н.

Проверила:

Андренко К.В.

Брест 2025

Цель: научиться конструировать нейросетевые классификаторы и выполнять их обучение на известных выборках компьютерного зрения.

Вариант 10.

Выборка: STL-10 (размеченная часть).
Размер исходного изображения: 96*96
Оптимизатор: Adam.

1. Выполнить конструирование своей модели СНС, обучить ее на выборке по заданию (использовать **torchvision.datasets**). Предпочтение отдавать как можно более простым архитектурам, базирующимся на базовых типах слоев (сверточный, полносвязный, подвыборочный, слой нелинейного преобразования). Оценить эффективность обучения на тестовой выборке, построить график изменения ошибки (matplotlib);
2. Ознакомьтесь с state-of-the-art результатами для предлагаемых выборок (https://paperswithcode.com/task/image-classification). Сделать выводы о результатах обучения СНС из п. 1;

3. Реализовать визуализацию работы СНС из пункта 1 (выбор и подачу на архитектуру произвольного изображения с выводом результата);

4. Оформить отчет по выполненной работе, загрузить исходный код и отчет в соответствующий репозиторий на github.

In [ ]:

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
import numpy as np
```

In [113]:

```
from torchvision import transforms

train_transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.RandomResizedCrop(96, scale=(0.8, 1.0)),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))
])

test_transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))
])

train_loader = torch.utils.data.DataLoader(
    torchvision.datasets.STL10('/files/', split='train', folds=1, download=True,
                               transform=train_transform),
    batch_size=64, shuffle=True)

test_loader = torch.utils.data.DataLoader(
    torchvision.datasets.STL10('/files/', split='test', folds=1, download=True,
                               transform=test_transform),
    batch_size=64, shuffle=True)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

In [114]:

```
class NN(nn.Module):
    def __init__(self):
        super(NN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=1)
        self.bn1 = nn.BatchNorm2d(32)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)
        self.bn2 = nn.BatchNorm2d(64)
        self.pool = nn.MaxPool2d(2, 2)
        # self.dropout1 = nn.Dropout(0.1)

        self.conv3 = nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1)
        self.bn3 = nn.BatchNorm2d(128)
        self.conv4 = nn.Conv2d(128, 256, kernel_size=3, stride=1, padding=1)
        self.bn4 = nn.BatchNorm2d(256)
        # self.dropout2 = nn.Dropout(0.2)

        self.conv5 = nn.Conv2d(256, 512, kernel_size=3, stride=1, padding=1)
        self.bn5 = nn.BatchNorm2d(512)
        self.conv6 = nn.Conv2d(512, 512, kernel_size=3, stride=1, padding=1)
        self.bn6 = nn.BatchNorm2d(512)
        # self.dropout3 = nn.Dropout(0.1)

        self.global_avg_pool = nn.AdaptiveAvgPool2d((1, 1))
```

```
        self.fc1 = nn.Linear(512, 256)
        self.fc2 = nn.Linear(256, 128)
        self.fc3 = nn.Linear(128, 10)
        self.relu = nn.LeakyReLU()
        self.dropout4 = nn.Dropout(0.5)

    def forward(self, x):
        x = self.pool(self.relu(self.bn1(self.conv1(x))))
        x = self.pool(self.relu(self.bn2(self.conv2(x))))
        # x = self.dropout1(x)

        x = self.pool(self.relu(self.bn3(self.conv3(x))))
        x = self.pool(self.relu(self.bn4(self.conv4(x))))
        # x = self.dropout2(x)

        x = self.pool(self.relu(self.bn5(self.conv5(x))))
        x = self.pool(self.relu(self.bn6(self.conv6(x))))
        # x = self.dropout3(x)

        x = self.global_avg_pool(x)
        x = x.view(-1, 512)

        x = self.relu(self.fc1(x))
        x = self.dropout4(x)
        x = self.relu(self.fc2(x))
        x = self.fc3(x)

        return x

model = NN()
```

In [115]:

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.00003)
```

In [116]:

```
def train(model, loader, criterion, optimizer, device):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0

    for images, labels in loader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()

        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

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

    accuracy = 100 * correct / total
    return running_loss / len(loader), accuracy

def test(model, loader, criterion, device):
    model.eval()
    running_loss = 0.0
    correct = 0
    total = 0

    with torch.no_grad():
        for images, labels in loader:
```

```
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            loss = criterion(outputs, labels)
            running_loss += loss.item()

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

    accuracy = 100 * correct / total
    return running_loss / len(loader), accuracy
```

In [117]:

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = model.to(device)

train_losses = []
test_losses = []
train_accuracies = []
test_accuracies = []

num_epochs = 100

for epoch in range(num_epochs):
    train_loss, train_accuracy = train(model, train_loader, criterion, optimizer, device
)
    test_loss, test_accuracy = test(model, test_loader, criterion, device)

    train_losses.append(train_loss)
    test_losses.append(test_loss)
    train_accuracies.append(train_accuracy)
    test_accuracies.append(test_accuracy)

    print(f'Epoch {epoch+1}/{num_epochs}, Train Loss: {train_loss:.4f}, Train Accuracy: {
train_accuracy:.2f}%, '
          f'Test Loss: {test_loss:.4f}, Test Accuracy: {test_accuracy:.2f}%')
```

```
Epoch 1/100, Train Loss: 2.2743, Train Accuracy: 13.70%, Test Loss: 2.3016, Test Accuracy
: 10.04%
Epoch 2/100, Train Loss: 2.2081, Train Accuracy: 21.00%, Test Loss: 2.2265, Test Accuracy
: 23.31%
Epoch 3/100, Train Loss: 2.1602, Train Accuracy: 22.50%, Test Loss: 2.0950, Test Accuracy
: 28.14%
Epoch 4/100, Train Loss: 2.0936, Train Accuracy: 24.50%, Test Loss: 2.0226, Test Accuracy
: 28.80%
Epoch 5/100, Train Loss: 2.0355, Train Accuracy: 27.00%, Test Loss: 1.9462, Test Accuracy
: 34.01%
Epoch 6/100, Train Loss: 1.9910, Train Accuracy: 27.90%, Test Loss: 1.8786, Test Accuracy
: 35.24%
Epoch 7/100, Train Loss: 1.9190, Train Accuracy: 34.30%, Test Loss: 1.8194, Test Accuracy
: 36.14%
Epoch 8/100, Train Loss: 1.8716, Train Accuracy: 31.90%, Test Loss: 1.7578, Test Accuracy
: 39.09%
Epoch 9/100, Train Loss: 1.8236, Train Accuracy: 35.10%, Test Loss: 1.6979, Test Accuracy
: 41.23%
Epoch 10/100, Train Loss: 1.7695, Train Accuracy: 37.10%, Test Loss: 1.6722, Test Accurac
y: 40.79%
Epoch 11/100, Train Loss: 1.7012, Train Accuracy: 39.60%, Test Loss: 1.6099, Test Accurac
y: 42.55%
Epoch 12/100, Train Loss: 1.6775, Train Accuracy: 38.90%, Test Loss: 1.5788, Test Accurac
y: 45.09%
Epoch 13/100, Train Loss: 1.6165, Train Accuracy: 42.50%, Test Loss: 1.5522, Test Accurac
y: 45.66%
Epoch 14/100, Train Loss: 1.6017, Train Accuracy: 41.30%, Test Loss: 1.5178, Test Accurac
y: 45.74%
Epoch 15/100, Train Loss: 1.5592, Train Accuracy: 43.00%, Test Loss: 1.5186, Test Accurac
y: 44.77%
Epoch 16/100, Train Loss: 1.5298, Train Accuracy: 45.10%, Test Loss: 1.5113, Test Accurac
y: 46.21%
Epoch 17/100, Train Loss: 1.5019, Train Accuracy: 45.60%, Test Loss: 1.4579, Test Accurac
y: 47.69%
```

```
Epoch 18/100, Train Loss: 1.4765, Train Accuracy: 47.30%, Test Loss: 1.4398, Test Accurac
y: 48.14%
Epoch 19/100, Train Loss: 1.4430, Train Accuracy: 47.90%, Test Loss: 1.4662, Test Accurac
y: 47.38%
Epoch 20/100, Train Loss: 1.4056, Train Accuracy: 50.20%, Test Loss: 1.4032, Test Accurac
y: 49.06%
Epoch 21/100, Train Loss: 1.3656, Train Accuracy: 50.30%, Test Loss: 1.4452, Test Accurac
y: 46.39%
Epoch 22/100, Train Loss: 1.3847, Train Accuracy: 49.40%, Test Loss: 1.3860, Test Accurac
y: 49.23%
Epoch 23/100, Train Loss: 1.3344, Train Accuracy: 52.20%, Test Loss: 1.3852, Test Accurac
y: 48.55%
Epoch 24/100, Train Loss: 1.2814, Train Accuracy: 53.80%, Test Loss: 1.3503, Test Accurac
y: 49.64%
Epoch 25/100, Train Loss: 1.2741, Train Accuracy: 54.70%, Test Loss: 1.3498, Test Accurac
y: 50.46%
Epoch 26/100, Train Loss: 1.2313, Train Accuracy: 56.60%, Test Loss: 1.3201, Test Accurac
y: 51.25%
Epoch 27/100, Train Loss: 1.2511, Train Accuracy: 56.50%, Test Loss: 1.3526, Test Accurac
y: 50.99%
Epoch 28/100, Train Loss: 1.2025, Train Accuracy: 57.20%, Test Loss: 1.3112, Test Accurac
y: 51.33%
Epoch 29/100, Train Loss: 1.1551, Train Accuracy: 61.10%, Test Loss: 1.2968, Test Accurac
y: 52.56%
Epoch 30/100, Train Loss: 1.1251, Train Accuracy: 60.50%, Test Loss: 1.2858, Test Accurac
y: 52.80%
Epoch 31/100, Train Loss: 1.1293, Train Accuracy: 61.80%, Test Loss: 1.2513, Test Accurac
y: 53.94%
Epoch 32/100, Train Loss: 1.0751, Train Accuracy: 64.60%, Test Loss: 1.2418, Test Accurac
y: 54.06%
Epoch 33/100, Train Loss: 1.0965, Train Accuracy: 60.80%, Test Loss: 1.2569, Test Accurac
y: 53.34%
Epoch 34/100, Train Loss: 1.0706, Train Accuracy: 63.30%, Test Loss: 1.2326, Test Accurac
y: 54.65%
Epoch 35/100, Train Loss: 1.0269, Train Accuracy: 65.00%, Test Loss: 1.2449, Test Accurac
y: 54.14%
Epoch 36/100, Train Loss: 1.0039, Train Accuracy: 65.30%, Test Loss: 1.1951, Test Accurac
y: 56.59%
Epoch 37/100, Train Loss: 0.9901, Train Accuracy: 66.50%, Test Loss: 1.2094, Test Accurac
y: 56.02%
Epoch 38/100, Train Loss: 0.9855, Train Accuracy: 67.40%, Test Loss: 1.2605, Test Accurac
y: 53.81%
Epoch 39/100, Train Loss: 0.9542, Train Accuracy: 69.70%, Test Loss: 1.2410, Test Accurac
y: 54.56%
Epoch 40/100, Train Loss: 0.9256, Train Accuracy: 68.20%, Test Loss: 1.1748, Test Accurac
y: 56.77%
Epoch 41/100, Train Loss: 0.8865, Train Accuracy: 69.60%, Test Loss: 1.2017, Test Accurac
y: 56.30%
Epoch 42/100, Train Loss: 0.8975, Train Accuracy: 70.40%, Test Loss: 1.2456, Test Accurac
y: 54.46%
Epoch 43/100, Train Loss: 0.8848, Train Accuracy: 71.10%, Test Loss: 1.1492, Test Accurac
y: 58.74%
Epoch 44/100, Train Loss: 0.8544, Train Accuracy: 70.70%, Test Loss: 1.2504, Test Accurac
y: 54.91%
Epoch 45/100, Train Loss: 0.8366, Train Accuracy: 72.00%, Test Loss: 1.1626, Test Accurac
y: 57.88%
Epoch 46/100, Train Loss: 0.8217, Train Accuracy: 72.90%, Test Loss: 1.2054, Test Accurac
y: 56.76%
Epoch 47/100, Train Loss: 0.8027, Train Accuracy: 74.40%, Test Loss: 1.1529, Test Accurac
y: 58.75%
Epoch 48/100, Train Loss: 0.7676, Train Accuracy: 75.20%, Test Loss: 1.1577, Test Accurac
y: 58.19%
Epoch 49/100, Train Loss: 0.7610, Train Accuracy: 75.10%, Test Loss: 1.1631, Test Accurac
y: 57.99%
Epoch 50/100, Train Loss: 0.7847, Train Accuracy: 74.20%, Test Loss: 1.1579, Test Accurac
y: 58.04%
Epoch 51/100, Train Loss: 0.7210, Train Accuracy: 77.30%, Test Loss: 1.2373, Test Accurac
y: 55.98%
Epoch 52/100, Train Loss: 0.7384, Train Accuracy: 75.70%, Test Loss: 1.1618, Test Accurac
y: 58.23%
Epoch 53/100, Train Loss: 0.7185, Train Accuracy: 76.70%, Test Loss: 1.1631, Test Accurac
y: 58.61%
```

```
Epoch 54/100, Train Loss: 0.7062, Train Accuracy: 77.40%, Test Loss: 1.1857, Test Accurac
y: 57.77%
Epoch 55/100, Train Loss: 0.6884, Train Accuracy: 78.60%, Test Loss: 1.1799, Test Accurac
y: 58.56%
Epoch 56/100, Train Loss: 0.6882, Train Accuracy: 77.90%, Test Loss: 1.1330, Test Accurac
y: 59.75%
Epoch 57/100, Train Loss: 0.6375, Train Accuracy: 80.90%, Test Loss: 1.2338, Test Accurac
y: 56.90%
Epoch 58/100, Train Loss: 0.6419, Train Accuracy: 80.90%, Test Loss: 1.1585, Test Accurac
y: 58.75%
Epoch 59/100, Train Loss: 0.6339, Train Accuracy: 80.20%, Test Loss: 1.1752, Test Accurac
y: 58.66%
Epoch 60/100, Train Loss: 0.6133, Train Accuracy: 80.90%, Test Loss: 1.1830, Test Accurac
y: 59.09%
Epoch 61/100, Train Loss: 0.5758, Train Accuracy: 82.30%, Test Loss: 1.1433, Test Accurac
y: 60.14%
Epoch 62/100, Train Loss: 0.5664, Train Accuracy: 82.60%, Test Loss: 1.1998, Test Accurac
y: 59.15%
Epoch 63/100, Train Loss: 0.5646, Train Accuracy: 82.30%, Test Loss: 1.1561, Test Accurac
y: 59.26%
Epoch 64/100, Train Loss: 0.5597, Train Accuracy: 83.10%, Test Loss: 1.2595, Test Accurac
y: 57.39%
Epoch 65/100, Train Loss: 0.5212, Train Accuracy: 84.10%, Test Loss: 1.1440, Test Accurac
y: 60.21%
Epoch 66/100, Train Loss: 0.5071, Train Accuracy: 85.20%, Test Loss: 1.1739, Test Accurac
y: 59.48%
Epoch 67/100, Train Loss: 0.4855, Train Accuracy: 85.10%, Test Loss: 1.1507, Test Accurac
y: 60.36%
Epoch 68/100, Train Loss: 0.5062, Train Accuracy: 84.20%, Test Loss: 1.1972, Test Accurac
y: 58.79%
Epoch 69/100, Train Loss: 0.4794, Train Accuracy: 86.00%, Test Loss: 1.1468, Test Accurac
y: 60.39%
Epoch 70/100, Train Loss: 0.4641, Train Accuracy: 87.40%, Test Loss: 1.2114, Test Accurac
y: 59.50%
Epoch 71/100, Train Loss: 0.4893, Train Accuracy: 84.90%, Test Loss: 1.2092, Test Accurac
y: 59.16%
Epoch 72/100, Train Loss: 0.4466, Train Accuracy: 86.20%, Test Loss: 1.2072, Test Accurac
y: 60.01%
Epoch 73/100, Train Loss: 0.4356, Train Accuracy: 87.00%, Test Loss: 1.2111, Test Accurac
y: 59.34%
Epoch 74/100, Train Loss: 0.4133, Train Accuracy: 88.30%, Test Loss: 1.2133, Test Accurac
y: 59.65%
Epoch 75/100, Train Loss: 0.4228, Train Accuracy: 87.20%, Test Loss: 1.2425, Test Accurac
y: 59.41%
Epoch 76/100, Train Loss: 0.4151, Train Accuracy: 88.60%, Test Loss: 1.2763, Test Accurac
y: 58.35%
Epoch 77/100, Train Loss: 0.4087, Train Accuracy: 87.80%, Test Loss: 1.1782, Test Accurac
y: 61.12%
Epoch 78/100, Train Loss: 0.4053, Train Accuracy: 87.50%, Test Loss: 1.2756, Test Accurac
y: 58.92%
Epoch 79/100, Train Loss: 0.3789, Train Accuracy: 89.10%, Test Loss: 1.2455, Test Accurac
y: 59.45%
Epoch 80/100, Train Loss: 0.3968, Train Accuracy: 88.50%, Test Loss: 1.2935, Test Accurac
y: 57.38%
Epoch 81/100, Train Loss: 0.3547, Train Accuracy: 91.20%, Test Loss: 1.2564, Test Accurac
y: 60.38%
Epoch 82/100, Train Loss: 0.3620, Train Accuracy: 88.90%, Test Loss: 1.2921, Test Accurac
y: 58.83%
Epoch 83/100, Train Loss: 0.3511, Train Accuracy: 90.20%, Test Loss: 1.2240, Test Accurac
y: 60.12%
Epoch 84/100, Train Loss: 0.3478, Train Accuracy: 89.60%, Test Loss: 1.2884, Test Accurac
y: 58.89%
Epoch 85/100, Train Loss: 0.2962, Train Accuracy: 93.40%, Test Loss: 1.3445, Test Accurac
y: 58.66%
Epoch 86/100, Train Loss: 0.3305, Train Accuracy: 90.90%, Test Loss: 1.2106, Test Accurac
y: 60.61%
Epoch 87/100, Train Loss: 0.3299, Train Accuracy: 89.60%, Test Loss: 1.2826, Test Accurac
y: 59.09%
Epoch 88/100, Train Loss: 0.3205, Train Accuracy: 90.70%, Test Loss: 1.3072, Test Accurac
y: 59.23%
Epoch 89/100, Train Loss: 0.2980, Train Accuracy: 92.20%, Test Loss: 1.3001, Test Accurac
y: 60.14%
```

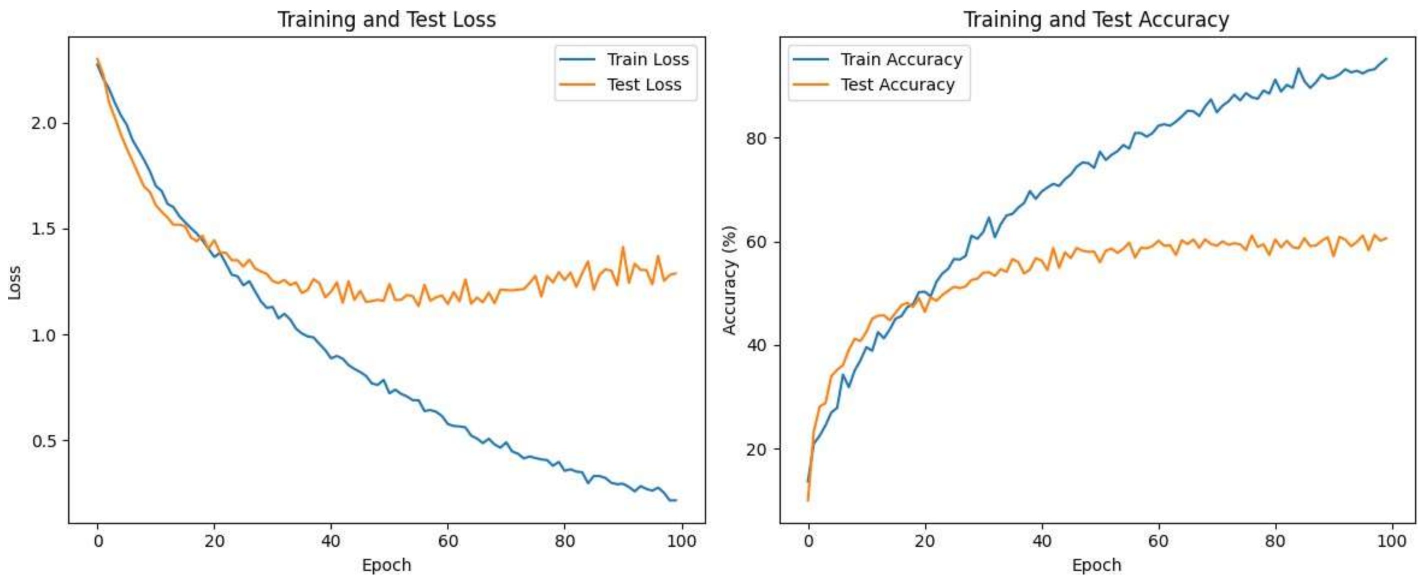
```
Epoch 90/100, Train Loss: 0.2913, Train Accuracy: 91.40%, Test Loss: 1.2319, Test Accurac
y: 60.81%
Epoch 91/100, Train Loss: 0.2936, Train Accuracy: 91.60%, Test Loss: 1.4124, Test Accurac
y: 57.14%
Epoch 92/100, Train Loss: 0.2785, Train Accuracy: 92.20%, Test Loss: 1.2426, Test Accurac
y: 60.88%
Epoch 93/100, Train Loss: 0.2576, Train Accuracy: 93.20%, Test Loss: 1.3336, Test Accurac
y: 60.34%
Epoch 94/100, Train Loss: 0.2824, Train Accuracy: 92.60%, Test Loss: 1.3045, Test Accurac
y: 59.06%
Epoch 95/100, Train Loss: 0.2692, Train Accuracy: 92.90%, Test Loss: 1.3033, Test Accurac
y: 59.90%
Epoch 96/100, Train Loss: 0.2609, Train Accuracy: 92.40%, Test Loss: 1.2357, Test Accurac
y: 61.12%
Epoch 97/100, Train Loss: 0.2752, Train Accuracy: 93.00%, Test Loss: 1.3697, Test Accurac
y: 58.30%
Epoch 98/100, Train Loss: 0.2513, Train Accuracy: 93.20%, Test Loss: 1.2528, Test Accurac
y: 61.26%
Epoch 99/100, Train Loss: 0.2147, Train Accuracy: 94.30%, Test Loss: 1.2809, Test Accurac
y: 60.14%
Epoch 100/100, Train Loss: 0.2152, Train Accuracy: 95.20%, Test Loss: 1.2874, Test Accura
cy: 60.59%
```

In [118]:

```
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(train_losses, label='Train Loss')
plt.plot(test_losses, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Test Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(train_accuracies, label='Train Accuracy')
plt.plot(test_accuracies, label='Test Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy (%)')
plt.title('Training and Test Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```



In [119]:

```
def imshow(img):
    img = img / 2 + 0.5
    np_img = img.numpy()
    plt.imshow(np.transpose(np_img, (1, 2, 0)))
```

```
    plt.axis('off')
    plt.show()

def test_random_image(model, loader, device):
    model.eval()
    images, labels = next(iter(loader))
    images, labels = images.to(device), labels.to(device)

    import random
    index = random.randint(0, images.size(0) - 1)
    image = images[index].unsqueeze(0)
    label = labels[index].item()

    output = model(image)
    _, predicted = torch.max(output, 1)
    predicted = predicted.item()

    imshow(image.cpu().squeeze())
    print(f'Predicted: {predicted}, Actual: {label}')

test_random_image(model, test_loader, device)
```

```
Predicted: 1, Actual: 1
```