How to use:

- The notebook is too heavy to run on Google Colab, so we ran it on our computer.
- Make sure you are running with CPU and not GPU (the code doesn't support it).
- Need to download ex3_308167675_211388491.zip file to your computer.
- Extract the files.
- Now you should have a folder with 2 folders and python notebook file:
    - checkpoints (saved models)
        - M1_+_SVM
        - DCGAN
        - WGAN
    - data (the dataset)
    - ex3_308167675_211388491.ipynb
    - readme.pdf (this file)
- Open the python notebook with an environment like Visual Studio Code on the containing folder.
- Click on Run All to train the models. For just the results see instructions below.
- The python notebook contains implementation of both sections 3 and 4.
- For each question it imports modules, inits the models, trains and evaluates the results.
- You can skip the training cells and run everything else to get the results – meaning run all cells but the cells of the main functions for question 3 and the 2 main functions of the DCGAN model and WGAN model.

Getting results for question 3:

Run all cells in order but the "Main" cell:

```
######################## Main ##########################
results = []
for dataset in ["FashionMNIST", "MNIST"]:
    for n in N_labeled:
        # Initialize M1 model, Transductive SVM, optimizer, and data loaders
        m1_model = M1(input_dim, hidden_dim, latent_dim)
        svm = SVM(kernel)
        optimizer = torch.optim.RMSprop(m1_model.parameters(), lr=learning_rate, momentum=momentum)

        ul_train_loader, l_train_loader, test_loader = load_data(labeled_size=n, batch_size=batch_size, dataset=dataset)
        # Train M1 model
        train_m1_model(m1_model, optimizer, ul_train_loader, device, num_epochs, log_interval)
        torch.save(m1_model.state_dict(), f'./checkpoints/M1_+_SVM/{dataset}/M1_{n}labeled.pth')

        # Extract features from M1 model for SVM
        X_train_features, y_train_labels = extract_m1_features(m1_model, l_train_loader, device)

        # Train SVM
        train_svm(svm, X_train_features, y_train_labels)
        with open(f'./checkpoints/M1_+_SVM/{dataset}/svm_{n}labeled.pkl','wb') as f:
            pickle.dump(svm,f)

        # Evaluate SVM
        correct, total = evaluate_svm(m1_model, svm, test_loader, device)
        accuracy = 100 * correct / total
        print(f'Accuracy of the SVM on the test images with {n} labeled images: {accuracy}%')

        results.append([dataset, n, accuracy, round(100 - accuracy,2)])
```

Getting results for question 4:

Run all cells in order, but the 2 “Main” cells:

1. DCGAN Main cell:

```
###################### DCGAN Main ######################
def main():
    train_dataloader = load_data(batch_size=64, dataset="FashionMNIST", show=False)

    generator, discriminator = init_gen_dis()

    criterion, discriminator_optim, generator_optim = init_optim(generator,
                                                                 discriminator,
                                                                 1e-4,
                                                                 beta1)

    # Create batch of latent vectors that we will use to visualize
    # the progression of the generator
    fixed_noise = torch.randn(64, z_latent_vector_size, 1, 1, device=device)

    G_losses, D_losses = train_dcgan(train_dataloader,
                                     generator,
                                     discriminator,
                                     criterion,
                                     discriminator_optim,
                                     generator_optim,
                                     fixed_noise)

    plot_losses(G_losses, D_losses)

main()
```

2. WGAN Main cell:

```
######################## WGAN Main ########################
def main():
    train_loader, test_loader = get_data_loader('fashion-mnist')

    model = WGAN_GP(channels=1,
                    generator_iters=7000,
                    critic_iters=5,
                    lambda_term=10,
                    learning_rate=1e-4,
                    b1=0.5,
                    b2=0.999,
                    batch_size=64)

    G_losses, D_losses = model.train(train_loader)

    plot_losses(G_losses, D_losses, "WGAN")

main()
```