Connected to ml_nn_labs (Python 3.9.18)

In [ ]:
```
try:
    del sys.modules["modules"]
    from modules import PrintLayer
except KeyError:
    from modules import (
        PrintLayer,
        ConvolutionalAutoencoder,
    )


def train(model, train_loader, criterion, optimizer):
    model.train()
    total_loss = 0
    for input in train_loader:
        input = input.to(device)
        input: torch.Tensor = input.permute(0, 3, 1, 2)
        output = model(input)

        loss = criterion(input, output)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    mean_loss = total_loss / len(train_loader)
    return mean_loss


def test(model, data_loader, criterion):
    model.train()
    total_loss = 0
```

```
    for input in data_loader:
        input = input.to(device)
        input: torch.Tensor = input.permute(0, 3, 1, 2)
        output = model(input)

        loss = criterion(input, output)
        total_loss += loss.item()

    mean_loss = total_loss / len(data_loader)
    return mean_loss
```

In [ ]:
```
from tqdm import tqdm
import torch.optim as optim

model = ConvolutionalAutoencoder(
    hidden_feature_dim_1=16,
    hidden_feature_dim_2=32,
    hidden_feature_dim_3=64,
    latent_dim=4, # OBS 4 for single 8 for multiple
).to(device)
criterion = nn.MSELoss()

initial_train_loss = test(model, train_loader, criterion)
print(f"{initial_train_loss=}")
initial_test_loss = test(model, test_loader, criterion)
print(f"{initial_test_loss=}")
```

```
initial_train_loss=21.88849560623169
initial_test_loss=21.685628593158416
```

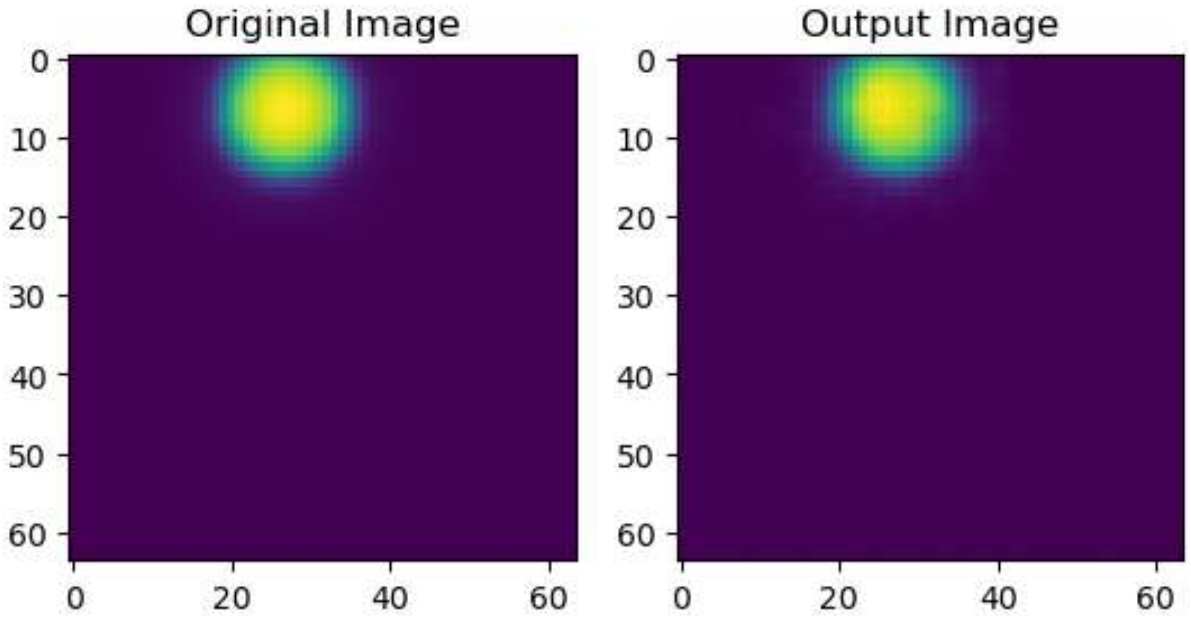
In [ ]:
```
num_epochs = 10

optimizer = optim.Adam(model.parameters(), lr=0.01)
for i, epoch in enumerate(tqdm(range(num_epochs))):
    train_loss = train(model, train_loader, criterion, optimizer)
    test_loss = test(model, test_loader, criterion)
```

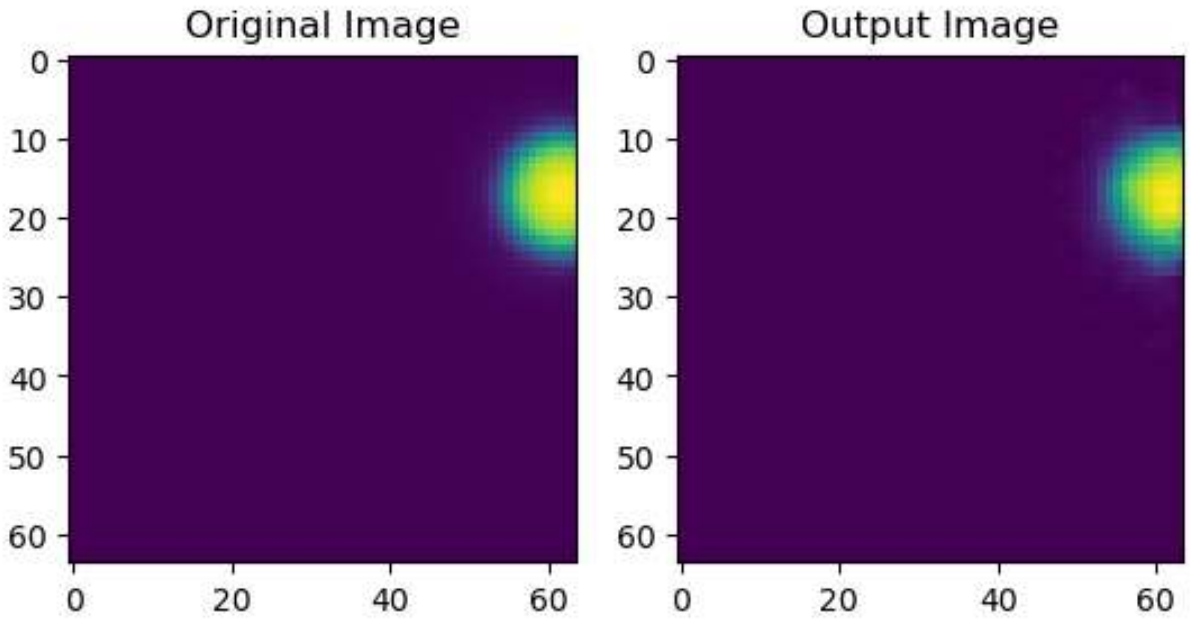
```
    print(f"Epoch {i+1}: Train loss {train_loss}, Test Loss {test_loss}")
```

```
 10%|▉         | 1/10 [00:11<01:40, 11.20s/it]
Epoch 1: Train loss 3.073351765871048, Test Loss 0.6255739915389984
 20%|██        | 2/10 [00:23<01:33, 11.73s/it]
Epoch 2: Train loss 0.5400525931239128, Test Loss 0.42144127216297217
 30%|███       | 3/10 [00:34<01:21, 11.63s/it]
Epoch 3: Train loss 0.412888870549202, Test Loss 0.40351634032238787
 40%|████      | 4/10 [00:49<01:16, 12.77s/it]
Epoch 4: Train loss 0.3943416316270828, Test Loss 0.3945157309404958
 50%|█████     | 5/10 [01:00<01:01, 12.25s/it]
Epoch 5: Train loss 0.362956719648838, Test Loss 0.4154529908594613
 60%|██████    | 6/10 [01:11<00:47, 11.90s/it]
Epoch 6: Train loss 0.30784215133190157, Test Loss 0.423424046498518
 70%|███████   | 7/10 [01:24<00:36, 12.01s/it]
Epoch 7: Train loss 0.20986419349312782, Test Loss 0.21640993573795111
 80%|████████  | 8/10 [01:36<00:24, 12.15s/it]
Epoch 8: Train loss 0.15568192343115805, Test Loss 0.1297685030621652
 90%|█████████ | 9/10 [01:48<00:11, 11.94s/it]
Epoch 9: Train loss 0.18246818803846837, Test Loss 0.08480721604995453
100%|██████████| 10/10 [01:59<00:00, 11.92s/it]
Epoch 10: Train loss 0.12445489001274108, Test Loss 0.09216466475837527
```

In [ ]:
```
for test_batch in test_loader:
    img = test_batch[torch.randint(low=0, high=31, size=(1,)).item(), :, :, :]

    output_img = model.forward(img.permute(2, 0, 1).unsqueeze(0).to(device)).cpu()
    output_img = output_img.detach().squeeze(0).permute(1, 2, 0)
    figure = plt.figure()
    subplot1 = figure.add_subplot(1, 2, 1)
    subplot1.imshow(img)
    subplot1.set_title("Original Image")
```

```
    subplot2 = figure.add_subplot(1, 2, 2)
    subplot2.imshow(output_img)
    subplot2.set_title("Output Image")

    plt.show()

    break
```





In [ ]:
```
import dill
torch.save(model, "models/convolutional_autoencoder_4000.pth", pickle_module=dill)
# torch.save(model, "models/multiple_particle_convolutional_autoencoder.pth")
```

In [ ]:
```
latent_dimensions = [1, 2, 3, 4, 8]
final_loss_thresholds = [20, 15, 10, 5, 5]

models = []
final_losses = []
for l, latent_dimension in tqdm(enumerate(latent_dimensions)):
    while True:
        model = ConvolutionalAutoencoder(
            hidden_feature_dim_1=16,
            hidden_feature_dim_2=32,
            hidden_feature_dim_3=64,
            latent_dim=latent_dimension,
        ).to(device)
```

```
        criterion = nn.MSELoss()
        optimizer = optim.Adam(model.parameters(), lr=0.01)

        num_epochs = 10

        for i, epoch in enumerate(tqdm(range(num_epochs))):
            train_loss = train(model, train_loader, criterion, optimizer)
            print(f"Epoch {i+1}: Train loss {train_loss}")
        test_loss = test(model, test_loader, criterion)
        if test_loss < final_loss_thresholds[l]:
            break

    print(f"{test_loss=}")

    models.append(model)
    final_losses.append(test_loss)
```

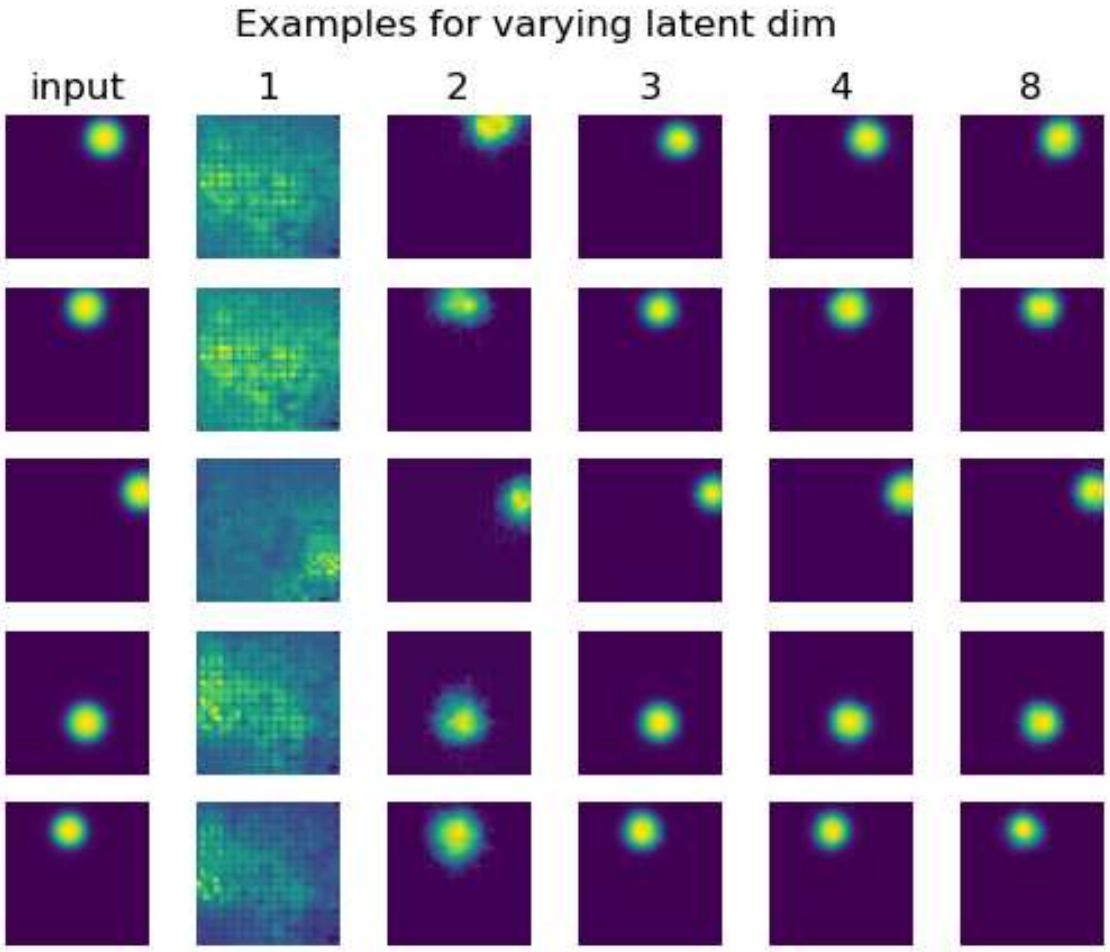
```
0it [00:00, ?it/s]
Epoch 1: Train loss 15.903006267166138
Epoch 2: Train loss 14.083030747222901
Epoch 3: Train loss 13.476288988113403
Epoch 4: Train loss 13.134857095336914
Epoch 5: Train loss 12.87836269378662
Epoch 6: Train loss 12.687604027938843
Epoch 7: Train loss 12.545945371246338
Epoch 8: Train loss 12.482487684249877
Epoch 9: Train loss 12.180882063293456
100%|██████████| 10/10 [01:36<00:00,  9.61s/it]
Epoch 10: Train loss 11.983132551956176
1it [01:37, 97.78s/it]
test_loss=11.500084664494086
Epoch 1: Train loss 12.114615138244629
```

```
Epoch 2: Train loss 6.92542668838501
Epoch 3: Train loss 5.347239723777771
Epoch 4: Train loss 5.180052912712097
Epoch 5: Train loss 4.980600651168824
Epoch 6: Train loss 4.836881259918213
Epoch 7: Train loss 6.565879984092712
Epoch 8: Train loss 4.743058064651489
Epoch 9: Train loss 4.615497487258911
100%|██████████| 10/10 [02:30<00:00, 15.02s/it]
Epoch 10: Train loss 4.934222789287567
2it [04:09, 129.28s/it]
test_loss=4.15368622979417
Epoch 1: Train loss 3.4441639117479323
Epoch 2: Train loss 0.5456375211119652
Epoch 3: Train loss 0.44859784348011017
Epoch 4: Train loss 0.39937861120700835
Epoch 5: Train loss 0.3759259365558624
Epoch 6: Train loss 0.37187664242982865
Epoch 7: Train loss 0.33483418200016024
Epoch 8: Train loss 0.3296351886510849
Epoch 9: Train loss 0.3282986355185509
100%|██████████| 10/10 [02:13<00:00, 13.39s/it]
Epoch 10: Train loss 0.3139798404932022
3it [06:24, 132.01s/it]
test_loss=0.2680701565342589
Epoch 1: Train loss 3.0218526768922804
Epoch 2: Train loss 0.578317249417305
Epoch 3: Train loss 0.409788971722126
Epoch 4: Train loss 0.2576785484433174
```

```
Epoch 5: Train loss 0.22755027964711189
Epoch 6: Train loss 0.16553410867452623
Epoch 7: Train loss 0.1808034847408533
Epoch 8: Train loss 0.16574874330759048
Epoch 9: Train loss 0.16334975450336933
100%|██████████| 10/10 [01:51<00:00, 11.19s/it]
Epoch 10: Train loss 0.14565541735291482
4it [08:17, 124.54s/it]
test_loss=0.1179842200999062
Epoch 1: Train loss 3.0715991183280944
Epoch 2: Train loss 0.41345072369575503
Epoch 3: Train loss 0.22537042288780212
Epoch 4: Train loss 0.20109877617955207
Epoch 5: Train loss 0.2718293482303619
Epoch 6: Train loss 0.13870088109970094
Epoch 7: Train loss 0.13629705036878587
Epoch 8: Train loss 0.11786837577223777
Epoch 9: Train loss 0.5128203569114208
100%|██████████| 10/10 [02:00<00:00, 12.10s/it]
Epoch 10: Train loss 0.1934864916741848
5it [10:19, 123.93s/it]
test_loss=0.11046332750505151
```

In [ ]:
```python
test_batch = next(iter(test_loader))

num_inputs = 5
input_images = test_batch[torch.randint(low=0, high=31, size=(num_inputs,)), :, :, :]


model_output_image_sets = []
for i, model in enumerate(models):
```

```
    out_images = model(input_images.permute(0, 3, 1, 2).to(device)).detach().cpu()
    model_output_image_sets.append(out_images)

figure = plt.figure()
figure.suptitle("Examples for varying latent dim")
plt.axis("off")
for i in range(num_inputs):
    # Plot input images
    subplot = figure.add_subplot(num_inputs, len(models) + 1, i * (len(models) + 1) + 1)
    subplot.imshow(input_images[i])
    subplot.axis("off")

    if i == 0:
        subplot.set_title("input")

    # Plot output images
    for j in range(len(models)):
        subplot = figure.add_subplot(
            num_inputs, len(models) + 1, i * (len(models) + 1) + j + 2
        )
        subplot.imshow(model_output_image_sets[j][i].permute(1, 2, 0))
        subplot.axis("off")

        if i == 0:
            subplot.set_title(f"{latent_dimensions[j]}")

plt.show()
```


Examples for varying latent dim
input
1
2
3
4
8