test_latent_reconstruction

March 7, 2025

1 Testing penultimate layer reconstruction

This notebook tests the reconstruction quality of data decoded from the penultimate layer of a classification model. The various setups are as follows: - Standard classifier, with decoder trained on penultimate layer (classifier weights frozen, only trained on cross-entropy loss) - Joint Training, with a joint loss for the classifier of reconstruction and cross-entropy loss (and of course only reconstruction loss for decoder) - Separate training of encoder and classification layer, where the encoder is trained only to minimise reconstruciton loss (i.e. a classifier layer is trained on the latent space of an autoencoder) - Supervised VAE, with a joint loss function and variational inference/latent regularisation

Set this to True if in Drive

[18]:
```
inDrive = True
```

[19]:
```
if inDrive:
    from google.colab import drive
    drive.mount('/content/drive')
    import os
    os.chdir('/content/drive/My Drive/Hybrid-CLUE/MyImplementation/
 ↪testing_notebooks')
    import sys

    # Add the parent directory to the system path
    current_dir = os.getcwd()
    parent_dir = os.path.dirname(current_dir)
    sys.path.insert(0, parent_dir)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[20]:
```
training = True
```

1.1 Setup

Import libraries

[29]:
```
import importlib
import models.regene_models as regene_models
```

```
importlib.reload(regene_models)
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import numpy as np
import os
```

Set the device

[22]:
```
if torch.cuda.is_available():
    device = torch.device("cuda")
elif torch.backends.mps.is_available():
    device = torch.device("mps")
else:
    device = torch.device("cpu")

print(f"Using device: {device}")
```

```
Using device: cuda
```

Load the Datasets

[30]:
```
import torch
from torch.utils.data import random_split
import torchvision
import torchvision.transforms as transforms

# Load the MNIST dataset
transform = transforms.Compose([transforms.ToTensor()])
trainset = torchvision.datasets.MNIST(root='../data', train=True,␣
 ↪download=True, transform=transform)
testset = torchvision.datasets.MNIST(root='../data', train=False,␣
 ↪download=True, transform=transform)

# Define the split ratio (e.g., 80% train, 20% validation)
train_ratio = 0.8
val_ratio = 1 - train_ratio

# Calculate the sizes of the splits
train_size = int(train_ratio * len(trainset))
val_size = len(trainset) - train_size

# Perform the split
train_subset, val_subset = random_split(trainset, [train_size, val_size])
```

```
# Create data loaders for the splits
trainloader = torch.utils.data.DataLoader(train_subset, batch_size=64,␣
 ↪shuffle=True, num_workers=2)
valloader = torch.utils.data.DataLoader(val_subset, batch_size=64,␣
 ↪shuffle=False, num_workers=2)
testloader = torch.utils.data.DataLoader(testset, batch_size=64, shuffle=False,␣
 ↪num_workers=2)
```

Set the latent dimension

[34]:
```
latent_dim = 256
epochs = 50
```

Create a models directory if it doesn't exist

[35]:
```
# Create models directory in parent directory if it doesn't exist
os.makedirs(os.path.join('..', 'model_saves', 'new_regene_models'),␣
 ↪exist_ok=True)
model_saves_path = os.path.join('..', 'model_saves', 'new_regene_models')
```

1.2 Classifier-dominated training

1.2.1 Classifier Training

First we define the classifier

[36]:
```
classifier = regene_models.Classifier(latent_dim=latent_dim, num_classes=10,␣
 ↪device=device)
```

Then we train

[37]:
```
# Train classifier and save
if training:
    classifier.train_classifier(trainloader, val_loader=valloader,␣
 ↪num_epochs=epochs, lr=0.001, patience=10, model_saves_dir=model_saves_path)
```

```
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/50], Train Loss: 0.1151
Epoch [1/50], Val Loss: 0.0632
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_classifier_256.pt
Epoch [2/50], Train Loss: 0.0443
Epoch [2/50], Val Loss: 0.0463
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_classifier_256.pt
Epoch [3/50], Train Loss: 0.0341
```

```
Epoch [3/50], Val Loss: 0.0549
Epoch [4/50], Train Loss: 0.0265
Epoch [4/50], Val Loss: 0.0487
Epoch [5/50], Train Loss: 0.0237
Epoch [5/50], Val Loss: 0.0461
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_classifier_256.pt
Epoch [6/50], Train Loss: 0.0173
Epoch [6/50], Val Loss: 0.0319
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_classifier_256.pt
Epoch [7/50], Train Loss: 0.0155
Epoch [7/50], Val Loss: 0.0472
Epoch [8/50], Train Loss: 0.0138
Epoch [8/50], Val Loss: 0.0365
Epoch [9/50], Train Loss: 0.0126
Epoch [9/50], Val Loss: 0.0401
Epoch [10/50], Train Loss: 0.0030
Epoch [10/50], Val Loss: 0.0331
Epoch [11/50], Train Loss: 0.0014
Epoch [11/50], Val Loss: 0.0295
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_classifier_256.pt
Epoch [12/50], Train Loss: 0.0018
Epoch [12/50], Val Loss: 0.0369
Epoch [13/50], Train Loss: 0.0037
Epoch [13/50], Val Loss: 0.0372
Epoch [14/50], Train Loss: 0.0027
Epoch [14/50], Val Loss: 0.0345
Epoch [15/50], Train Loss: 0.0007
Epoch [15/50], Val Loss: 0.0339
Epoch [16/50], Train Loss: 0.0003
Epoch [16/50], Val Loss: 0.0388
Epoch [17/50], Train Loss: 0.0001
Epoch [17/50], Val Loss: 0.0343
Epoch [18/50], Train Loss: 0.0000
Epoch [18/50], Val Loss: 0.0340
Epoch [19/50], Train Loss: 0.0000
Epoch [19/50], Val Loss: 0.0347
Epoch [20/50], Train Loss: 0.0000
Epoch [20/50], Val Loss: 0.0353
Epoch [21/50], Train Loss: 0.0000
Epoch [21/50], Val Loss: 0.0356
Early stopping triggered after 21 epochs
```

1.2.2 Classifier Loading

Load the classifier

[ ]:
```
classifier.load_state_dict(torch.load(os.path.join(model_saves_path,␣
↪'classifier.pth'), map_location=device))
```

```
/var/folders/tb/ccwl9r592hn9v_xpq9s1bzlr0000gn/T/ipykernel_13002/1365348556.py:1
: FutureWarning: You are using `torch.load` with `weights_only=False` (the
current default value), which uses the default pickle module implicitly. It is
possible to construct malicious pickle data which will execute arbitrary code
during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  classifier.load_state_dict(torch.load(os.path.join(model_saves_path,
'classifier.pth'), map_location=device))
```

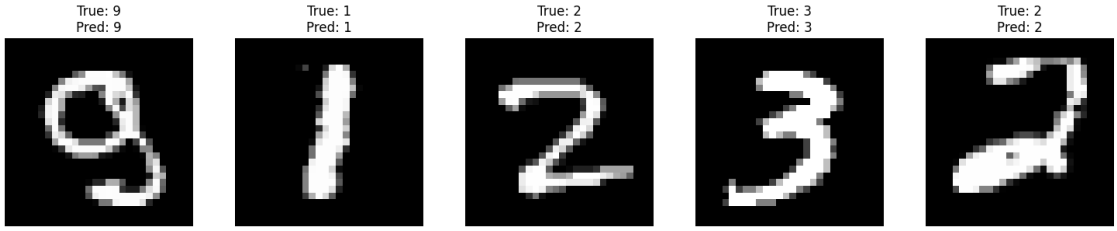
[ ]:
```
<All keys matched successfully>
```

1.2.3 Classifier Testing

First let's test the classifier on a few images

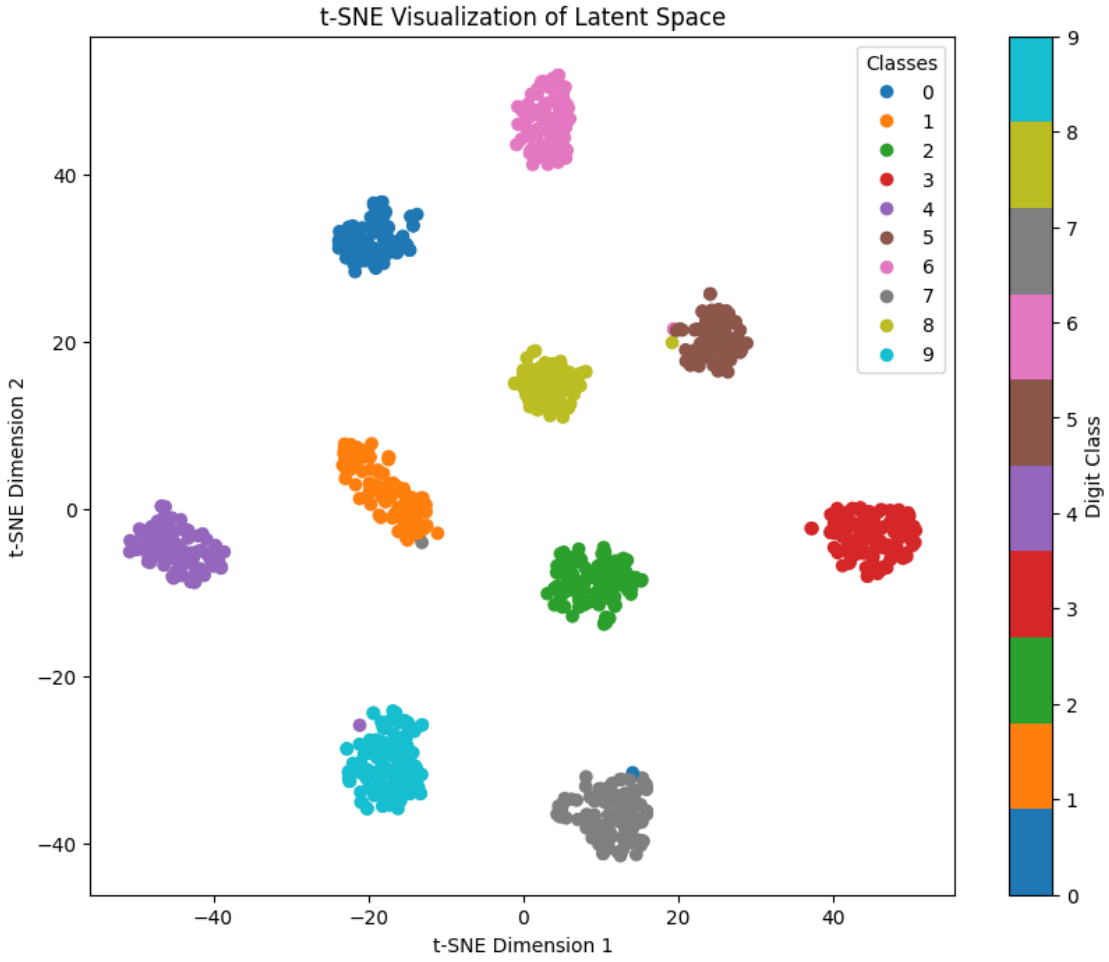
[38]:
```
# Get random indices for test images
random_indices = torch.randint(0, len(testset), (5,))
images = torch.stack([testset[i][0] for i in random_indices])
labels = torch.tensor([testset[i][1] for i in random_indices])

# Get predictions
classifier.eval()  # Set to evaluation mode
with torch.no_grad():
    images = images.to(device)
    _, predictions = classifier(images)
    predicted_classes = torch.argmax(predictions, dim=1)

# Plot images with true and predicted labels
plt.figure(figsize=(15, 3))
for i in range(5):
    plt.subplot(1, 5, i + 1)
    plt.imshow(images[i].cpu().squeeze().numpy(), cmap='gray')
    plt.title(f'True: {labels[i].item()}\nPred: {predicted_classes[i].cpu().
 ↪item()}')
    plt.axis('off')

plt.tight_layout()
```

```
plt.show()
```



We'll also visualise the latent space. This is done by taking the latent representations of 50 training images and plotting them in 2D using t-SNE.

[39]:
```
# Get latent representations for 50 random training images
random_indices = torch.randint(0, len(testset), (1000,))
images = torch.stack([testset[i][0] for i in random_indices])
labels = torch.tensor([testset[i][1] for i in random_indices])

# Get latent representations
classifier.eval()
with torch.no_grad():
    images = images.to(device)
    latent_reps, _ = classifier(images)
    latent_reps = latent_reps.cpu().numpy()

# Perform t-SNE dimensionality reduction
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2, random_state=42)
latent_2d = tsne.fit_transform(latent_reps)

# Plot the 2D latent space
plt.figure(figsize=(10, 8))
scatter = plt.scatter(latent_2d[:, 0], latent_2d[:, 1], c=labels, cmap='tab10')
plt.colorbar(scatter, label='Digit Class')
plt.title('t-SNE Visualization of Latent Space')
plt.xlabel('t-SNE Dimension 1')
plt.ylabel('t-SNE Dimension 2')
plt.legend(*scatter.legend_elements(), title="Classes")
plt.show()
```



1.2.4 Decoder Training

We define the decoder, and then train it using the classifier's latent space.

[40]:
```
decoder = regene_models.Decoder(latent_dim=latent_dim, device=device)
```

[43]:
```
if training:
    decoder.train_decoder(trainloader, classifier, num_epochs=epochs, lr=0.001,␣
 ↪model_saves_dir=model_saves_path, patience=10, val_loader=valloader,)
```

```
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Decoder Epoch [1/50], Train Loss: 0.0164
Decoder Epoch [1/50], Val Loss: 0.0158
Saved best model to:
```

```
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [2/50], Train Loss: 0.0154
Decoder Epoch [2/50], Val Loss: 0.0151
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [3/50], Train Loss: 0.0147
Decoder Epoch [3/50], Val Loss: 0.0147
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [4/50], Train Loss: 0.0143
Decoder Epoch [4/50], Val Loss: 0.0144
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [5/50], Train Loss: 0.0138
Decoder Epoch [5/50], Val Loss: 0.0143
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [6/50], Train Loss: 0.0135
Decoder Epoch [6/50], Val Loss: 0.0143
Decoder Epoch [7/50], Train Loss: 0.0132
Decoder Epoch [7/50], Val Loss: 0.0137
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [8/50], Train Loss: 0.0130
Decoder Epoch [8/50], Val Loss: 0.0136
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [9/50], Train Loss: 0.0128
Decoder Epoch [9/50], Val Loss: 0.0134
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [10/50], Train Loss: 0.0126
Decoder Epoch [10/50], Val Loss: 0.0132
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [11/50], Train Loss: 0.0124
Decoder Epoch [11/50], Val Loss: 0.0130
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [12/50], Train Loss: 0.0122
Decoder Epoch [12/50], Val Loss: 0.0130
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [13/50], Train Loss: 0.0121
Decoder Epoch [13/50], Val Loss: 0.0131
Decoder Epoch [14/50], Train Loss: 0.0119
Decoder Epoch [14/50], Val Loss: 0.0128
Saved best model to:
```

```
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [15/50], Train Loss: 0.0118
Decoder Epoch [15/50], Val Loss: 0.0127
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [16/50], Train Loss: 0.0117
Decoder Epoch [16/50], Val Loss: 0.0127
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [17/50], Train Loss: 0.0116
Decoder Epoch [17/50], Val Loss: 0.0127
Decoder Epoch [18/50], Train Loss: 0.0115
Decoder Epoch [18/50], Val Loss: 0.0127
Decoder Epoch [19/50], Train Loss: 0.0114
Decoder Epoch [19/50], Val Loss: 0.0125
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [20/50], Train Loss: 0.0113
Decoder Epoch [20/50], Val Loss: 0.0126
Decoder Epoch [21/50], Train Loss: 0.0112
Decoder Epoch [21/50], Val Loss: 0.0123
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [22/50], Train Loss: 0.0111
Decoder Epoch [22/50], Val Loss: 0.0126
Decoder Epoch [23/50], Train Loss: 0.0110
Decoder Epoch [23/50], Val Loss: 0.0123
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [24/50], Train Loss: 0.0110
Decoder Epoch [24/50], Val Loss: 0.0123
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [25/50], Train Loss: 0.0109
Decoder Epoch [25/50], Val Loss: 0.0123
Decoder Epoch [26/50], Train Loss: 0.0108
Decoder Epoch [26/50], Val Loss: 0.0124
Decoder Epoch [27/50], Train Loss: 0.0108
Decoder Epoch [27/50], Val Loss: 0.0122
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [28/50], Train Loss: 0.0107
Decoder Epoch [28/50], Val Loss: 0.0122
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [29/50], Train Loss: 0.0106
Decoder Epoch [29/50], Val Loss: 0.0121
Saved best model to:
```

```
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [30/50], Train Loss: 0.0106
Decoder Epoch [30/50], Val Loss: 0.0123
Decoder Epoch [31/50], Train Loss: 0.0105
Decoder Epoch [31/50], Val Loss: 0.0120
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [32/50], Train Loss: 0.0104
Decoder Epoch [32/50], Val Loss: 0.0121
Decoder Epoch [33/50], Train Loss: 0.0104
Decoder Epoch [33/50], Val Loss: 0.0121
Decoder Epoch [34/50], Train Loss: 0.0104
Decoder Epoch [34/50], Val Loss: 0.0120
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [35/50], Train Loss: 0.0103
Decoder Epoch [35/50], Val Loss: 0.0121
Decoder Epoch [36/50], Train Loss: 0.0103
Decoder Epoch [36/50], Val Loss: 0.0120
Decoder Epoch [37/50], Train Loss: 0.0102
Decoder Epoch [37/50], Val Loss: 0.0121
Decoder Epoch [38/50], Train Loss: 0.0095
Decoder Epoch [38/50], Val Loss: 0.0115
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [39/50], Train Loss: 0.0094
Decoder Epoch [39/50], Val Loss: 0.0115
Decoder Epoch [40/50], Train Loss: 0.0094
Decoder Epoch [40/50], Val Loss: 0.0115
Decoder Epoch [41/50], Train Loss: 0.0094
Decoder Epoch [41/50], Val Loss: 0.0115
Decoder Epoch [42/50], Train Loss: 0.0090
Decoder Epoch [42/50], Val Loss: 0.0113
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [43/50], Train Loss: 0.0089
Decoder Epoch [43/50], Val Loss: 0.0113
Decoder Epoch [44/50], Train Loss: 0.0089
Decoder Epoch [44/50], Val Loss: 0.0113
Decoder Epoch [45/50], Train Loss: 0.0089
Decoder Epoch [45/50], Val Loss: 0.0113
Decoder Epoch [46/50], Train Loss: 0.0087
Decoder Epoch [46/50], Val Loss: 0.0112
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
Decoder Epoch [47/50], Train Loss: 0.0087
Decoder Epoch [47/50], Val Loss: 0.0112
Decoder Epoch [48/50], Train Loss: 0.0087
```

```
Decoder Epoch [48/50], Val Loss: 0.0112
Decoder Epoch [49/50], Train Loss: 0.0087
Decoder Epoch [49/50], Val Loss: 0.0112
Decoder Epoch [50/50], Train Loss: 0.0086
Decoder Epoch [50/50], Val Loss: 0.0111
Saved best model to:
../model_saves/new_regene_models/classifier_dominated_decoder_256.pt
```

1.2.5 Decoder Loading

Load the decoder

[ ]:
```
decoder.load_state_dict(torch.load(os.path.join(model_saves_path, 'decoder.
↪pth'), map_location=device))
```

```
/var/folders/tb/ccwl9r592hn9v_xpq9s1bzlr0000gn/T/ipykernel_13002/1833794902.py:1
: FutureWarning: You are using `torch.load` with `weights_only=False` (the
current default value), which uses the default pickle module implicitly. It is
possible to construct malicious pickle data which will execute arbitrary code
during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  decoder.load_state_dict(torch.load(os.path.join(model_saves_path,
'decoder.pth'), map_location=device))
```

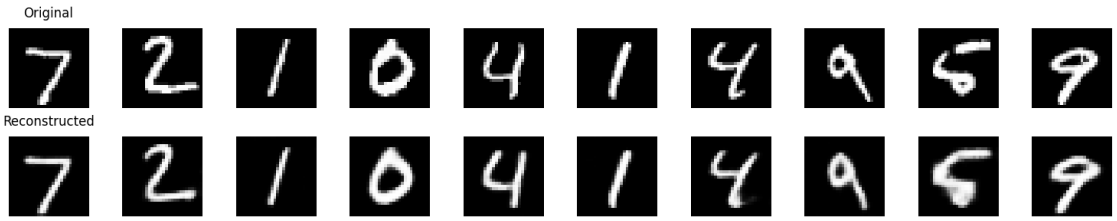
[ ]:
```
<All keys matched successfully>
```

1.2.6 Decoder Testing

First less visualise some reconstructions

[44]:
```
# Get 10 random images from training set
dataiter = iter(testloader)
images, _ = next(dataiter)
images = images[:10].to(device)

# Get reconstructions
classifier.eval()
decoder.eval()
with torch.no_grad():
    z, _ = classifier(images)
    reconstructed = decoder(z)
```

```
# Plot original vs reconstructed images
fig, axes = plt.subplots(2, 10, figsize=(15, 3))
for i in range(10):
    # Original images
    axes[0,i].imshow(images[i].cpu().squeeze(), cmap='gray')
    axes[0,i].axis('off')
    if i == 0:
        axes[0,i].set_title('Original', pad=10)

    # Reconstructed images
    axes[1,i].imshow(reconstructed[i].cpu().squeeze(), cmap='gray')
    axes[1,i].axis('off')
    if i == 0:
        axes[1,i].set_title('Reconstructed', pad=10)

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



1.3 Joint training

1.3.1 Training

Train the models

Let's try training the models with a joint objective

[45]:
```
from importlib import reload
import models.regene_models as regene_models
importlib.reload(regene_models)
from models.regene_models import Classifier
```

Alpha determines how much weight is given to the reconstruction loss.

[51]:
```
joint_decoder = regene_models.Decoder(latent_dim=latent_dim, device=device)
joint_classifier = regene_models.Classifier(latent_dim=latent_dim,␣
 ↪num_classes=10, device=device)

if training:
```

```
    regene_models.train_joint(joint_classifier, joint_decoder, trainloader,␣
↪num_epochs=epochs, lr=0.001, lambda_recon=0.8, val_loader=valloader,␣
↪patience=10, model_saves_dir=model_saves_path)
```

```
Epoch [1/50], Train Total Loss: 0.0517, Train Classification Loss: 0.1152, Train
Reconstruction Loss: 0.0358
Epoch [1/50], Val Total Loss: 0.0236, Val Classification Loss: 0.0436, Val
Reconstruction Loss: 0.0185
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [2/50], Train Total Loss: 0.0205, Train Classification Loss: 0.0418, Train
Reconstruction Loss: 0.0152
Epoch [2/50], Val Total Loss: 0.0187, Val Classification Loss: 0.0418, Val
Reconstruction Loss: 0.0130
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [3/50], Train Total Loss: 0.0167, Train Classification Loss: 0.0344, Train
Reconstruction Loss: 0.0123
Epoch [3/50], Val Total Loss: 0.0153, Val Classification Loss: 0.0301, Val
Reconstruction Loss: 0.0116
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [4/50], Train Total Loss: 0.0136, Train Classification Loss: 0.0251, Train
Reconstruction Loss: 0.0107
Epoch [4/50], Val Total Loss: 0.0161, Val Classification Loss: 0.0366, Val
Reconstruction Loss: 0.0110
Epoch [5/50], Train Total Loss: 0.0119, Train Classification Loss: 0.0204, Train
Reconstruction Loss: 0.0098
Epoch [5/50], Val Total Loss: 0.0167, Val Classification Loss: 0.0459, Val
Reconstruction Loss: 0.0094
Epoch [6/50], Train Total Loss: 0.0106, Train Classification Loss: 0.0172, Train
Reconstruction Loss: 0.0090
Epoch [6/50], Val Total Loss: 0.0157, Val Classification Loss: 0.0394, Val
Reconstruction Loss: 0.0098
Epoch [7/50], Train Total Loss: 0.0073, Train Classification Loss: 0.0065, Train
Reconstruction Loss: 0.0075
Epoch [7/50], Val Total Loss: 0.0114, Val Classification Loss: 0.0279, Val
Reconstruction Loss: 0.0073
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [8/50], Train Total Loss: 0.0064, Train Classification Loss: 0.0043, Train
Reconstruction Loss: 0.0070
Epoch [8/50], Val Total Loss: 0.0109, Val Classification Loss: 0.0266, Val
Reconstruction Loss: 0.0070
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [9/50], Train Total Loss: 0.0063, Train Classification Loss: 0.0047, Train
Reconstruction Loss: 0.0067
```

```
Epoch [9/50], Val Total Loss: 0.0121, Val Classification Loss: 0.0323, Val
Reconstruction Loss: 0.0070
Epoch [10/50], Train Total Loss: 0.0068, Train Classification Loss: 0.0073,
Train Reconstruction Loss: 0.0067
Epoch [10/50], Val Total Loss: 0.0141, Val Classification Loss: 0.0438, Val
Reconstruction Loss: 0.0067
Epoch [11/50], Train Total Loss: 0.0057, Train Classification Loss: 0.0038,
Train Reconstruction Loss: 0.0062
Epoch [11/50], Val Total Loss: 0.0122, Val Classification Loss: 0.0335, Val
Reconstruction Loss: 0.0068
Epoch [12/50], Train Total Loss: 0.0048, Train Classification Loss: 0.0014,
Train Reconstruction Loss: 0.0057
Epoch [12/50], Val Total Loss: 0.0097, Val Classification Loss: 0.0258, Val
Reconstruction Loss: 0.0057
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [13/50], Train Total Loss: 0.0044, Train Classification Loss: 0.0005,
Train Reconstruction Loss: 0.0054
Epoch [13/50], Val Total Loss: 0.0098, Val Classification Loss: 0.0272, Val
Reconstruction Loss: 0.0054
Epoch [14/50], Train Total Loss: 0.0042, Train Classification Loss: 0.0002,
Train Reconstruction Loss: 0.0051
Epoch [14/50], Val Total Loss: 0.0095, Val Classification Loss: 0.0270, Val
Reconstruction Loss: 0.0051
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [15/50], Train Total Loss: 0.0041, Train Classification Loss: 0.0006,
Train Reconstruction Loss: 0.0050
Epoch [15/50], Val Total Loss: 0.0105, Val Classification Loss: 0.0320, Val
Reconstruction Loss: 0.0051
Epoch [16/50], Train Total Loss: 0.0043, Train Classification Loss: 0.0015,
Train Reconstruction Loss: 0.0050
Epoch [16/50], Val Total Loss: 0.0094, Val Classification Loss: 0.0274, Val
Reconstruction Loss: 0.0049
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [17/50], Train Total Loss: 0.0038, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0047
Epoch [17/50], Val Total Loss: 0.0098, Val Classification Loss: 0.0303, Val
Reconstruction Loss: 0.0047
Epoch [18/50], Train Total Loss: 0.0041, Train Classification Loss: 0.0019,
Train Reconstruction Loss: 0.0047
Epoch [18/50], Val Total Loss: 0.0104, Val Classification Loss: 0.0324, Val
Reconstruction Loss: 0.0049
Epoch [19/50], Train Total Loss: 0.0040, Train Classification Loss: 0.0017,
Train Reconstruction Loss: 0.0046
Epoch [19/50], Val Total Loss: 0.0100, Val Classification Loss: 0.0308, Val
Reconstruction Loss: 0.0047
```

```
Epoch [20/50], Train Total Loss: 0.0036, Train Classification Loss: 0.0005,
Train Reconstruction Loss: 0.0044
Epoch [20/50], Val Total Loss: 0.0091, Val Classification Loss: 0.0275, Val
Reconstruction Loss: 0.0045
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [21/50], Train Total Loss: 0.0034, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0043
Epoch [21/50], Val Total Loss: 0.0091, Val Classification Loss: 0.0275, Val
Reconstruction Loss: 0.0045
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [22/50], Train Total Loss: 0.0033, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0042
Epoch [22/50], Val Total Loss: 0.0089, Val Classification Loss: 0.0275, Val
Reconstruction Loss: 0.0043
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [23/50], Train Total Loss: 0.0033, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0041
Epoch [23/50], Val Total Loss: 0.0091, Val Classification Loss: 0.0279, Val
Reconstruction Loss: 0.0043
Epoch [24/50], Train Total Loss: 0.0032, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0040
Epoch [24/50], Val Total Loss: 0.0088, Val Classification Loss: 0.0274, Val
Reconstruction Loss: 0.0041
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [25/50], Train Total Loss: 0.0032, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0039
Epoch [25/50], Val Total Loss: 0.0087, Val Classification Loss: 0.0270, Val
Reconstruction Loss: 0.0041
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [26/50], Train Total Loss: 0.0031, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0038
Epoch [26/50], Val Total Loss: 0.0087, Val Classification Loss: 0.0277, Val
Reconstruction Loss: 0.0040
Epoch [27/50], Train Total Loss: 0.0030, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0037
Epoch [27/50], Val Total Loss: 0.0088, Val Classification Loss: 0.0282, Val
Reconstruction Loss: 0.0039
Epoch [28/50], Train Total Loss: 0.0029, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0036
Epoch [28/50], Val Total Loss: 0.0088, Val Classification Loss: 0.0287, Val
Reconstruction Loss: 0.0038
Epoch [29/50], Train Total Loss: 0.0029, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0036
```

```
Epoch [29/50], Val Total Loss: 0.0088, Val Classification Loss: 0.0282, Val
Reconstruction Loss: 0.0039
Epoch [30/50], Train Total Loss: 0.0028, Train Classification Loss: 0.0000,
Train Reconstruction Loss: 0.0035
Epoch [30/50], Val Total Loss: 0.0086, Val Classification Loss: 0.0279, Val
Reconstruction Loss: 0.0038
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [31/50], Train Total Loss: 0.0028, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0034
Epoch [31/50], Val Total Loss: 0.0085, Val Classification Loss: 0.0278, Val
Reconstruction Loss: 0.0037
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [32/50], Train Total Loss: 0.0027, Train Classification Loss: 0.0001,
Train Reconstruction Loss: 0.0034
Epoch [32/50], Val Total Loss: 0.0084, Val Classification Loss: 0.0278, Val
Reconstruction Loss: 0.0036
Saved best models to: ../model_saves/new_regene_models/joint_classifier_256.pt
and ../model_saves/new_regene_models/joint_decoder_256.pt
Epoch [33/50], Train Total Loss: 0.0027, Train Classification Loss: 0.0000,
Train Reconstruction Loss: 0.0034
Epoch [33/50], Val Total Loss: 0.0086, Val Classification Loss: 0.0283, Val
Reconstruction Loss: 0.0036
Epoch [34/50], Train Total Loss: 0.0027, Train Classification Loss: 0.0000,
Train Reconstruction Loss: 0.0033
Epoch [34/50], Val Total Loss: 0.0085, Val Classification Loss: 0.0283, Val
Reconstruction Loss: 0.0035
Epoch [35/50], Train Total Loss: 0.0026, Train Classification Loss: 0.0000,
Train Reconstruction Loss: 0.0033
Epoch [35/50], Val Total Loss: 0.0085, Val Classification Loss: 0.0285, Val
Reconstruction Loss: 0.0035
Epoch [36/50], Train Total Loss: 0.0026, Train Classification Loss: 0.0000,
Train Reconstruction Loss: 0.0032
Epoch [36/50], Val Total Loss: 0.0086, Val Classification Loss: 0.0289, Val
Reconstruction Loss: 0.0035
Epoch [37/50], Train Total Loss: 0.0026, Train Classification Loss: 0.0000,
Train Reconstruction Loss: 0.0032
Epoch [37/50], Val Total Loss: 0.0085, Val Classification Loss: 0.0287, Val
Reconstruction Loss: 0.0034
Epoch [38/50], Train Total Loss: 0.0025, Train Classification Loss: 0.0000,
Train Reconstruction Loss: 0.0032
Epoch [38/50], Val Total Loss: 0.0085, Val Classification Loss: 0.0287, Val
Reconstruction Loss: 0.0034
Epoch [39/50], Train Total Loss: 0.0025, Train Classification Loss: 0.0000,
Train Reconstruction Loss: 0.0031
Epoch [39/50], Val Total Loss: 0.0085, Val Classification Loss: 0.0290, Val
Reconstruction Loss: 0.0034
```

```
Epoch [40/50], Train Total Loss: 0.0025, Train Classification Loss: 0.0000,
Train Reconstruction Loss: 0.0031
Epoch [40/50], Val Total Loss: 0.0085, Val Classification Loss: 0.0289, Val
Reconstruction Loss: 0.0034
Epoch [41/50], Train Total Loss: 0.0025, Train Classification Loss: 0.0000,
Train Reconstruction Loss: 0.0031
Epoch [41/50], Val Total Loss: 0.0084, Val Classification Loss: 0.0288, Val
Reconstruction Loss: 0.0033
Epoch [42/50], Train Total Loss: 0.0025, Train Classification Loss: 0.0000,
Train Reconstruction Loss: 0.0031
Epoch [42/50], Val Total Loss: 0.0085, Val Classification Loss: 0.0292, Val
Reconstruction Loss: 0.0033
Early stopping triggered after 42 epochs
```

1.3.2 Loading

Load the models

[ ]:
```
joint_classifier.load_state_dict(torch.load(os.path.join(model_saves_path,␣
↪'joint_classifier.pth'), map_location=device))
joint_decoder.load_state_dict(torch.load(os.path.join(model_saves_path,␣
↪'joint_decoder.pth'), map_location=device))
```

```
/var/folders/tb/ccwl9r592hn9v_xpq9s1bzlr0000gn/T/ipykernel_13002/3085312799.py:1
: FutureWarning: You are using `torch.load` with `weights_only=False` (the
current default value), which uses the default pickle module implicitly. It is
possible to construct malicious pickle data which will execute arbitrary code
during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  joint_classifier.load_state_dict(torch.load(os.path.join(model_saves_path,
'joint_classifier.pth'), map_location=device))
/var/folders/tb/ccwl9r592hn9v_xpq9s1bzlr0000gn/T/ipykernel_13002/3085312799.py:2
: FutureWarning: You are using `torch.load` with `weights_only=False` (the
current default value), which uses the default pickle module implicitly. It is
possible to construct malicious pickle data which will execute arbitrary code
during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
```

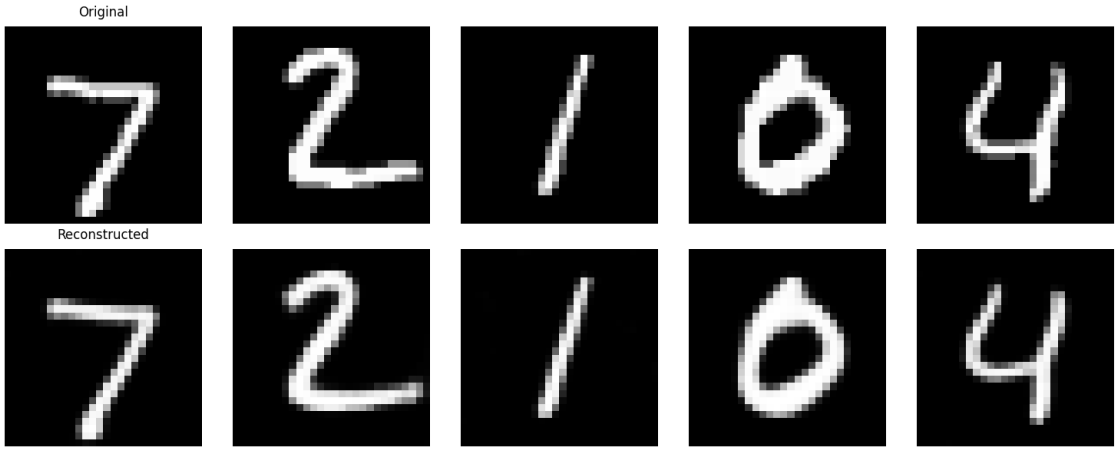
```
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  joint_decoder.load_state_dict(torch.load(os.path.join(model_saves_path,
'joint_decoder.pth'), map_location=device))
```

[ ]:
```
<All keys matched successfully>
```

1.3.3 Testing

Test the models

[47]:
```
# Get some test images
dataiter = iter(testloader)
images, labels = next(dataiter)
images = images.to(device)

# Get reconstructions
with torch.no_grad():
    z, _ = joint_classifier(images)
    reconstructed = joint_decoder(z)

# Plot original vs reconstructed images
fig, axes = plt.subplots(2, 5, figsize=(15, 6))
for i in range(5):
    # Original images
    axes[0,i].imshow(images[i].cpu().squeeze(), cmap='gray')
    axes[0,i].axis('off')
    if i == 0:
        axes[0,i].set_title('Original', pad=10)

    # Reconstructed images
    axes[1,i].imshow(reconstructed[i].cpu().squeeze(), cmap='gray')
    axes[1,i].axis('off')
    if i == 0:
        axes[1,i].set_title('Reconstructed', pad=10)

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



1.4 Training encoder and classifier separately

In this final section, we will train the encoder and classifier separately. The encoder is trained to minimise the reconstruction loss, and the classifier is trained to minimise the cross-entropy loss on the enocders latent space.

Define models

[52]:
```
from models.regene_models import train_autoencoder, train_classifier_only

separate_classifier = regene_models.Classifier(latent_dim=latent_dim,␣
↪num_classes=10, device=device)
separate_decoder = regene_models.Decoder(latent_dim=latent_dim, device=device)
```

1.4.1 Training

Train the autoencoder

[53]:
```
if training:
    # Train autoencoder
    train_autoencoder(classifier=separate_classifier, decoder=separate_decoder,␣
↪train_loader=trainloader, num_epochs=epochs, lr=0.001, val_loader=valloader,␣
↪patience = 10, model_saves_dir=model_saves_path)
```

```
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/50], Train Reconstruction Loss: 0.0280
Epoch [1/50], Val Reconstruction Loss: 0.0078
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
```

```
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [2/50], Train Reconstruction Loss: 0.0060
Epoch [2/50], Val Reconstruction Loss: 0.0052
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [3/50], Train Reconstruction Loss: 0.0045
Epoch [3/50], Val Reconstruction Loss: 0.0041
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [4/50], Train Reconstruction Loss: 0.0038
Epoch [4/50], Val Reconstruction Loss: 0.0037
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [5/50], Train Reconstruction Loss: 0.0035
Epoch [5/50], Val Reconstruction Loss: 0.0033
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [6/50], Train Reconstruction Loss: 0.0032
Epoch [6/50], Val Reconstruction Loss: 0.0029
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [7/50], Train Reconstruction Loss: 0.0030
Epoch [7/50], Val Reconstruction Loss: 0.0031
Epoch [8/50], Train Reconstruction Loss: 0.0028
Epoch [8/50], Val Reconstruction Loss: 0.0029
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [9/50], Train Reconstruction Loss: 0.0027
Epoch [9/50], Val Reconstruction Loss: 0.0025
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [10/50], Train Reconstruction Loss: 0.0025
Epoch [10/50], Val Reconstruction Loss: 0.0027
Epoch [11/50], Train Reconstruction Loss: 0.0025
Epoch [11/50], Val Reconstruction Loss: 0.0024
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [12/50], Train Reconstruction Loss: 0.0024
Epoch [12/50], Val Reconstruction Loss: 0.0023
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
```

```
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [13/50], Train Reconstruction Loss: 0.0023
Epoch [13/50], Val Reconstruction Loss: 0.0024
Epoch [14/50], Train Reconstruction Loss: 0.0022
Epoch [14/50], Val Reconstruction Loss: 0.0023
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [15/50], Train Reconstruction Loss: 0.0021
Epoch [15/50], Val Reconstruction Loss: 0.0021
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [16/50], Train Reconstruction Loss: 0.0021
Epoch [16/50], Val Reconstruction Loss: 0.0024
Epoch [17/50], Train Reconstruction Loss: 0.0021
Epoch [17/50], Val Reconstruction Loss: 0.0021
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [18/50], Train Reconstruction Loss: 0.0020
Epoch [18/50], Val Reconstruction Loss: 0.0021
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [19/50], Train Reconstruction Loss: 0.0019
Epoch [19/50], Val Reconstruction Loss: 0.0020
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [20/50], Train Reconstruction Loss: 0.0019
Epoch [20/50], Val Reconstruction Loss: 0.0020
Epoch [21/50], Train Reconstruction Loss: 0.0019
Epoch [21/50], Val Reconstruction Loss: 0.0019
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [22/50], Train Reconstruction Loss: 0.0018
Epoch [22/50], Val Reconstruction Loss: 0.0019
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [23/50], Train Reconstruction Loss: 0.0018
Epoch [23/50], Val Reconstruction Loss: 0.0020
Epoch [24/50], Train Reconstruction Loss: 0.0018
Epoch [24/50], Val Reconstruction Loss: 0.0019
Epoch [25/50], Train Reconstruction Loss: 0.0017
```

```
Epoch [25/50], Val Reconstruction Loss: 0.0018
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [26/50], Train Reconstruction Loss: 0.0017
Epoch [26/50], Val Reconstruction Loss: 0.0018
Epoch [27/50], Train Reconstruction Loss: 0.0017
Epoch [27/50], Val Reconstruction Loss: 0.0019
Epoch [28/50], Train Reconstruction Loss: 0.0017
Epoch [28/50], Val Reconstruction Loss: 0.0018
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [29/50], Train Reconstruction Loss: 0.0017
Epoch [29/50], Val Reconstruction Loss: 0.0018
Epoch [30/50], Train Reconstruction Loss: 0.0016
Epoch [30/50], Val Reconstruction Loss: 0.0018
Epoch [31/50], Train Reconstruction Loss: 0.0016
Epoch [31/50], Val Reconstruction Loss: 0.0018
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [32/50], Train Reconstruction Loss: 0.0016
Epoch [32/50], Val Reconstruction Loss: 0.0017
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [33/50], Train Reconstruction Loss: 0.0016
Epoch [33/50], Val Reconstruction Loss: 0.0018
Epoch [34/50], Train Reconstruction Loss: 0.0016
Epoch [34/50], Val Reconstruction Loss: 0.0017
Epoch [35/50], Train Reconstruction Loss: 0.0015
Epoch [35/50], Val Reconstruction Loss: 0.0017
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [36/50], Train Reconstruction Loss: 0.0015
Epoch [36/50], Val Reconstruction Loss: 0.0017
Epoch [37/50], Train Reconstruction Loss: 0.0015
Epoch [37/50], Val Reconstruction Loss: 0.0016
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [38/50], Train Reconstruction Loss: 0.0015
Epoch [38/50], Val Reconstruction Loss: 0.0017
Epoch [39/50], Train Reconstruction Loss: 0.0015
Epoch [39/50], Val Reconstruction Loss: 0.0016
Epoch [40/50], Train Reconstruction Loss: 0.0015
```

```
Epoch [40/50], Val Reconstruction Loss: 0.0016
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [41/50], Train Reconstruction Loss: 0.0015
Epoch [41/50], Val Reconstruction Loss: 0.0017
Epoch [42/50], Train Reconstruction Loss: 0.0015
Epoch [42/50], Val Reconstruction Loss: 0.0016
Epoch [43/50], Train Reconstruction Loss: 0.0014
Epoch [43/50], Val Reconstruction Loss: 0.0016
Epoch [44/50], Train Reconstruction Loss: 0.0013
Epoch [44/50], Val Reconstruction Loss: 0.0014
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [45/50], Train Reconstruction Loss: 0.0013
Epoch [45/50], Val Reconstruction Loss: 0.0014
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [46/50], Train Reconstruction Loss: 0.0013
Epoch [46/50], Val Reconstruction Loss: 0.0014
Epoch [47/50], Train Reconstruction Loss: 0.0013
Epoch [47/50], Val Reconstruction Loss: 0.0014
Saved best models to: ../model_saves/new_regene_models/autoencoder_dominated_cla
ssifier_encoder_only_256.pt and
../model_saves/new_regene_models/autoencoder_dominated_decoder_256.pt
Epoch [48/50], Train Reconstruction Loss: 0.0013
Epoch [48/50], Val Reconstruction Loss: 0.0014
Epoch [49/50], Train Reconstruction Loss: 0.0013
Epoch [49/50], Val Reconstruction Loss: 0.0014
Epoch [50/50], Train Reconstruction Loss: 0.0013
Epoch [50/50], Val Reconstruction Loss: 0.0014
```

Train the classifier layer

[54]:
```
if training:
    # Train autoencoder
    train_classifier_only(separate_classifier, trainloader, num_epochs=epochs,␣
 ↪lr=0.001, val_loader=valloader, patience=10,␣
 ↪model_saves_dir=model_saves_path)
```

```
Epoch [1/50], Train Loss: 0.5854
Epoch [1/50], Val Loss: 0.3041
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [2/50], Train Loss: 0.2714
Epoch [2/50], Val Loss: 0.2636
Saved best model to:
```

```
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [3/50], Train Loss: 0.2406
Epoch [3/50], Val Loss: 0.2468
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [4/50], Train Loss: 0.2236
Epoch [4/50], Val Loss: 0.2204
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [5/50], Train Loss: 0.2158
Epoch [5/50], Val Loss: 0.2053
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [6/50], Train Loss: 0.2079
Epoch [6/50], Val Loss: 0.2213
Epoch [7/50], Train Loss: 0.2020
Epoch [7/50], Val Loss: 0.2083
Epoch [8/50], Train Loss: 0.2004
Epoch [8/50], Val Loss: 0.2064
Epoch [9/50], Train Loss: 0.1843
Epoch [9/50], Val Loss: 0.1937
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [10/50], Train Loss: 0.1815
Epoch [10/50], Val Loss: 0.1909
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [11/50], Train Loss: 0.1812
Epoch [11/50], Val Loss: 0.1958
Epoch [12/50], Train Loss: 0.1788
Epoch [12/50], Val Loss: 0.1878
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [13/50], Train Loss: 0.1771
Epoch [13/50], Val Loss: 0.1902
Epoch [14/50], Train Loss: 0.1765
Epoch [14/50], Val Loss: 0.1963
Epoch [15/50], Train Loss: 0.1762
Epoch [15/50], Val Loss: 0.1870
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [16/50], Train Loss: 0.1756
Epoch [16/50], Val Loss: 0.1980
Epoch [17/50], Train Loss: 0.1738
Epoch [17/50], Val Loss: 0.1853
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [18/50], Train Loss: 0.1713
```

```
Epoch [18/50], Val Loss: 0.1793
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [19/50], Train Loss: 0.1725
Epoch [19/50], Val Loss: 0.1845
Epoch [20/50], Train Loss: 0.1701
Epoch [20/50], Val Loss: 0.1800
Epoch [21/50], Train Loss: 0.1711
Epoch [21/50], Val Loss: 0.1809
Epoch [22/50], Train Loss: 0.1645
Epoch [22/50], Val Loss: 0.1771
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [23/50], Train Loss: 0.1638
Epoch [23/50], Val Loss: 0.1737
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [24/50], Train Loss: 0.1638
Epoch [24/50], Val Loss: 0.1772
Epoch [25/50], Train Loss: 0.1625
Epoch [25/50], Val Loss: 0.1752
Epoch [26/50], Train Loss: 0.1622
Epoch [26/50], Val Loss: 0.1772
Epoch [27/50], Train Loss: 0.1597
Epoch [27/50], Val Loss: 0.1738
Epoch [28/50], Train Loss: 0.1587
Epoch [28/50], Val Loss: 0.1727
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [29/50], Train Loss: 0.1586
Epoch [29/50], Val Loss: 0.1731
Epoch [30/50], Train Loss: 0.1587
Epoch [30/50], Val Loss: 0.1720
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [31/50], Train Loss: 0.1586
Epoch [31/50], Val Loss: 0.1713
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [32/50], Train Loss: 0.1580
Epoch [32/50], Val Loss: 0.1714
Epoch [33/50], Train Loss: 0.1578
Epoch [33/50], Val Loss: 0.1739
Epoch [34/50], Train Loss: 0.1582
Epoch [34/50], Val Loss: 0.1740
Epoch [35/50], Train Loss: 0.1567
Epoch [35/50], Val Loss: 0.1709
Saved best model to:
```

```
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [36/50], Train Loss: 0.1565
Epoch [36/50], Val Loss: 0.1723
Epoch [37/50], Train Loss: 0.1567
Epoch [37/50], Val Loss: 0.1712
Epoch [38/50], Train Loss: 0.1563
Epoch [38/50], Val Loss: 0.1702
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [39/50], Train Loss: 0.1558
Epoch [39/50], Val Loss: 0.1708
Epoch [40/50], Train Loss: 0.1563
Epoch [40/50], Val Loss: 0.1717
Epoch [41/50], Train Loss: 0.1566
Epoch [41/50], Val Loss: 0.1699
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [42/50], Train Loss: 0.1564
Epoch [42/50], Val Loss: 0.1716
Epoch [43/50], Train Loss: 0.1558
Epoch [43/50], Val Loss: 0.1701
Epoch [44/50], Train Loss: 0.1557
Epoch [44/50], Val Loss: 0.1701
Epoch [45/50], Train Loss: 0.1553
Epoch [45/50], Val Loss: 0.1695
Saved best model to:
../model_saves/new_regene_models/autoencoder_dominated_classifier_full_256.pt
Epoch [46/50], Train Loss: 0.1544
Epoch [46/50], Val Loss: 0.1704
Epoch [47/50], Train Loss: 0.1552
Epoch [47/50], Val Loss: 0.1701
Epoch [48/50], Train Loss: 0.1553
Epoch [48/50], Val Loss: 0.1699
Epoch [49/50], Train Loss: 0.1547
Epoch [49/50], Val Loss: 0.1697
Epoch [50/50], Train Loss: 0.1543
Epoch [50/50], Val Loss: 0.1697
```

1.4.2 Loading

Load the models

[ ]:
```
separate_classifier.load_state_dict(torch.load(os.path.join(model_saves_path,␣
↪'separate_classifier.pth'), map_location=device))
separate_decoder.load_state_dict(torch.load(os.path.join(model_saves_path,␣
↪'separate_decoder.pth'), map_location=device))
```

```
/var/folders/tb/ccwl9r592hn9v_xpq9s1bzlr0000gn/T/ipykernel_13002/2821388973.py:1
: FutureWarning: You are using `torch.load` with `weights_only=False` (the
current default value), which uses the default pickle module implicitly. It is
possible to construct malicious pickle data which will execute arbitrary code
during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  separate_classifier.load_state_dict(torch.load(os.path.join(model_saves_path,
'separate_classifier.pth'), map_location=device))
/var/folders/tb/ccwl9r592hn9v_xpq9s1bzlr0000gn/T/ipykernel_13002/2821388973.py:2
: FutureWarning: You are using `torch.load` with `weights_only=False` (the
current default value), which uses the default pickle module implicitly. It is
possible to construct malicious pickle data which will execute arbitrary code
during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  separate_decoder.load_state_dict(torch.load(os.path.join(model_saves_path,
'separate_decoder.pth'), map_location=device))
```

[ ]:
```
<All keys matched successfully>
```

1.5 Supervised Variational Autoencoder

1.5.1 Setup

Define model

[55]:
```
from models.SVAE import SVAE

svae = SVAE(latent_dim=latent_dim, num_classes=10, device=device)
```

Train model

[ ]:
```
if training:
    svae.train_model(trainloader, num_epochs=5, lr=0.001, beta=0.01, alpha=10.0)
```

```
    # Save the trained SVAE model
    torch.save(svae.state_dict(), os.path.join(model_saves_path, 'svae_model.
 ↪pth'))
```

Load model

[57]:
```
svae.load_state_dict(torch.load(os.path.join('../model_saves', 'svae_model.
 ↪pth'), map_location=device))
```

```
<ipython-input-57-37a0b1cc4bd3>:1: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  svae.load_state_dict(torch.load(os.path.join('../model_saves',
'svae_model.pth'), map_location=device))
```

[57]:
```
<All keys matched successfully>
```

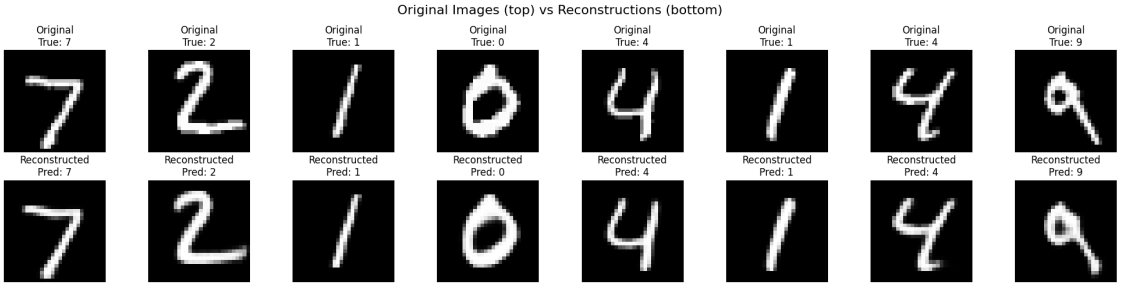
1.5.2 Testing

Test the model

First let's simply qualitatively test the model's ability to recosntruct and classify a few images.

[58]:
```
# Test SVAE reconstruction and classification
test_images, test_labels = next(iter(testloader))
test_images = test_images.to(device)
test_labels = test_labels.to(device)

# Get reconstructions and predictions
with torch.no_grad():
    # Change this line - the SVAE forward method returns (x_recon, y_pred, mu,␣
 ↪log_var)
    x_recon, y_pred, mu, log_var = svae(test_images)
    # Apply softmax before argmax
    pred_probs = torch.softmax(y_pred, dim=1)
    pred_labels = torch.argmax(pred_probs, dim=1)

# Convert tensors to numpy for plotting
test_images = test_images.cpu().numpy()
```

```
recon_batch = x_recon.cpu().numpy()
test_labels = test_labels.cpu().numpy()
pred_labels = pred_labels.cpu().numpy()

# Plot original and reconstructed images
fig, axes = plt.subplots(2, 8, figsize=(20, 5))
fig.suptitle('Original Images (top) vs Reconstructions (bottom)', fontsize=16)

for i in range(8):
    # Original images
    axes[0, i].imshow(test_images[i].squeeze(), cmap='gray')
    axes[0, i].set_title(f'Original\nTrue: {test_labels[i]}')
    axes[0, i].axis('off')

    # Reconstructed images
    axes[1, i].imshow(recon_batch[i].squeeze(), cmap='gray')
    axes[1, i].set_title(f'Reconstructed\nPred: {pred_labels[i]}')
    axes[1, i].axis('off')

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



Next, let's get a proper MSE reconstruction error and classification accuracy.

[59]:
```
# Calculate MSE loss and classification accuracy
mse_loss = nn.MSELoss()
total_mse = 0
total_correct = 0
total_samples = 0

with torch.no_grad():
    for images, labels in testloader:
        images = images.to(device)
        labels = labels.to(device)

        # Get model outputs
```

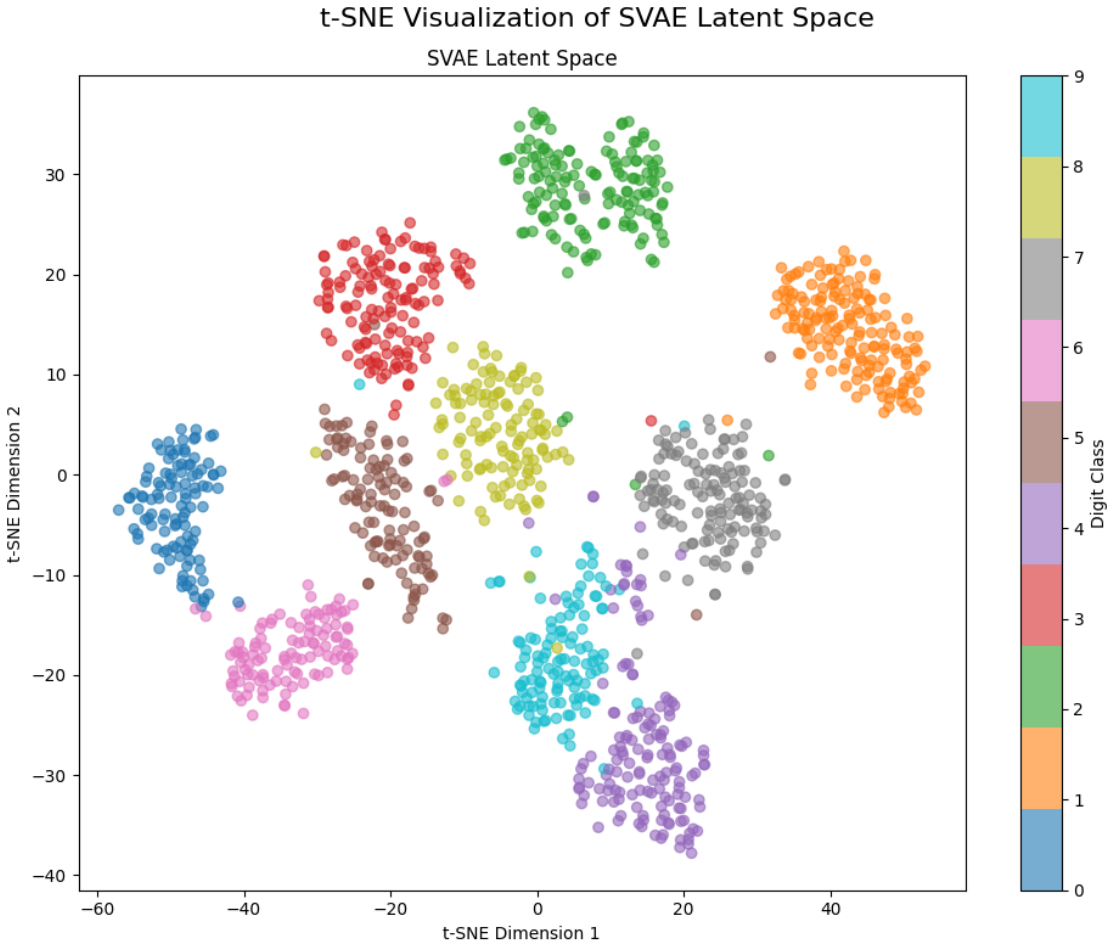
```
        recon, pred, mu, log_var = svae(images)

        # Calculate MSE
        total_mse += mse_loss(recon, images).item() * images.size(0)

        # Calculate accuracy
        pred_probs = torch.softmax(pred, dim=1)
        pred_labels = torch.argmax(pred_probs, dim=1)
        total_correct += (pred_labels == labels).sum().item()
        total_samples += labels.size(0)

# Calculate averages
avg_mse = total_mse / total_samples
accuracy = 100 * total_correct / total_samples

print(f"Test Set Results:")
print(f"Average MSE: {avg_mse:.4f}")
print(f"Classification Accuracy: {accuracy:.2f}%")
```

```
Test Set Results:
Average MSE: 0.0051
Classification Accuracy: 99.14%
```

Finally, we'll visualise the latent space of the SVAE.

[60]:
```
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE

# Get more images from trainloader
all_images = []
all_labels = []
num_batches = 20  # Increased from 5 to 20 batches of data

for i, (images, labels) in enumerate(testloader):
    if i >= num_batches:
        break
    all_images.append(images)
    all_labels.append(labels)

# Concatenate all batches
test_images_tensor = torch.cat(all_images, dim=0).to(device)
test_labels = torch.cat(all_labels, dim=0).cpu().numpy()

# Get latent representations for test data
with torch.no_grad():
    mu, log_var = svae.encode(test_images_tensor)
    z = svae.reparameterize(mu, log_var)
    latent_reps = z.cpu().numpy()
```

```
# Create the figure for t-SNE visualization
fig, ax = plt.subplots(figsize=(10, 8))
fig.suptitle('t-SNE Visualization of SVAE Latent Space', fontsize=16)

# Run t-SNE and plot
tsne = TSNE(n_components=2, random_state=42)
latent_2d = tsne.fit_transform(latent_reps)
scatter = ax.scatter(latent_2d[:, 0], latent_2d[:, 1], c=test_labels,␣
 ↪cmap='tab10', alpha=0.6)
ax.set_title('SVAE Latent Space')
ax.set_xlabel('t-SNE Dimension 1')
ax.set_ylabel('t-SNE Dimension 2')

# Add colorbar
plt.colorbar(scatter, label='Digit Class')

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



1.6 Comparison

We will now compare the performance of the different models.

First we need to create wrapper classes so that the SVAE can be used in the same way as the other models. This makes our testing loops more straightforward.

[61]:
```
# Create wrapper classes to make SVAE behave like classifier/decoder pairs
class SVAEClassifier(nn.Module):
    def __init__(self, svae):
        super().__init__()
        self.svae = svae

    def forward(self, x):
        # SVAE forward returns (x_recon, y_pred, mu, log_var)
        _, y_pred, mu, log_var = self.svae(x)
        z = self.svae.reparameterize(mu, log_var)
        return z, y_pred

class SVAEDecoder(nn.Module):
    def __init__(self, svae):
        super().__init__()
        self.svae = svae

    def forward(self, z):
        return self.svae.decode(z)

# Create wrapped SVAE models
svae_classifier = SVAEClassifier(svae).to(device)
svae_decoder = SVAEDecoder(svae).to(device)
```

[62]:
```
models = [(classifier, decoder), (joint_classifier, joint_decoder),␣
↪(separate_classifier, separate_decoder), (svae_classifier, svae_decoder)]
```

[66]:
```
%pip install torchmetrics
```

```
Collecting torchmetrics
  Downloading torchmetrics-1.6.2-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: numpy>1.20.0 in /usr/local/lib/python3.11/dist-
packages (from torchmetrics) (1.26.4)
Requirement already satisfied: packaging>17.1 in /usr/local/lib/python3.11/dist-
packages (from torchmetrics) (24.2)
Requirement already satisfied: torch>=2.0.0 in /usr/local/lib/python3.11/dist-
packages (from torchmetrics) (2.5.1+cu124)
Collecting lightning-utilities>=0.8.0 (from torchmetrics)
  Downloading lightning_utilities-0.14.0-py3-none-any.whl.metadata (5.6 kB)
```

```
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-
packages (from lightning-utilities>=0.8.0->torchmetrics) (75.1.0)
Requirement already satisfied: typing_extensions in
/usr/local/lib/python3.11/dist-packages (from lightning-
utilities>=0.8.0->torchmetrics) (4.12.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from torch>=2.0.0->torchmetrics) (3.17.0)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-
packages (from torch>=2.0.0->torchmetrics) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages
(from torch>=2.0.0->torchmetrics) (3.1.5)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages
(from torch>=2.0.0->torchmetrics) (2024.10.0)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=2.0.0->torchmetrics)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=2.0.0->torchmetrics)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=2.0.0->torchmetrics)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=2.0.0->torchmetrics)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=2.0.0->torchmetrics)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=2.0.0->torchmetrics)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=2.0.0->torchmetrics)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=2.0.0->torchmetrics)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=2.0.0->torchmetrics)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in
/usr/local/lib/python3.11/dist-packages (from torch>=2.0.0->torchmetrics)
(2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch>=2.0.0->torchmetrics)
(12.4.127)
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=2.0.0->torchmetrics)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
```

```
manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.1.0 in /usr/local/lib/python3.11/dist-
packages (from torch>=2.0.0->torchmetrics) (3.1.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-
packages (from torch>=2.0.0->torchmetrics) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from
sympy==1.13.1->torch>=2.0.0->torchmetrics) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from
jinja2->torch>=2.0.0->torchmetrics) (3.0.2)
Downloading torchmetrics-1.6.2-py3-none-any.whl (931 kB)
                         931.6/931.6 kB
49.7 MB/s eta 0:00:00
Downloading lightning_utilities-0.14.0-py3-none-any.whl (28 kB)
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4
MB)
                         363.4/363.4 MB
3.0 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (13.8 MB)
                         13.8/13.8 MB
105.4 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (24.6 MB)
                         24.6/24.6 MB
87.8 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (883 kB)
                         883.7/883.7 kB
50.5 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl
(664.8 MB)
                         664.8/664.8 MB
2.2 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl
(211.5 MB)
                         211.5/211.5 MB
4.8 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
manylinux2014_x86_64.whl (56.3 MB)
                         56.3/56.3 MB
39.6 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl (127.9 MB)
                         127.9/127.9 MB
18.1 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
```

```
manylinux2014_x86_64.whl (207.5 MB)
                         207.5/207.5 MB
4.9 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (21.1 MB)
                         21.1/21.1 MB
92.8 MB/s eta 0:00:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12,
nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-
cuda-cupti-cu12, nvidia-cublas-cu12, lightning-utilities, nvidia-cusparse-cu12,
nvidia-cudnn-cu12, nvidia-cusolver-cu12, torchmetrics
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
```

```
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed lightning-utilities-0.14.0 nvidia-cublas-cu12-12.4.5.8
nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-
runtime-cu12-12.4.127 nvidia-cudnn-cu12-9.1.0.70 nvidia-cufft-cu12-11.2.1.3
nvidia-curand-cu12-10.3.5.147 nvidia-cusolver-cu12-11.6.1.9 nvidia-cusparse-
cu12-12.3.1.170 nvidia-nvjitlink-cu12-12.4.127 torchmetrics-1.6.2
```

[68]:
```python
import torch.nn.functional as F
from torchmetrics import Accuracy
import pandas as pd
from IPython.display import display

# Function to calculate metrics
def calculate_metrics(classifier, decoder, test_loader):
    classifier.eval()
    decoder.eval()

    total = 0
    correct = 0
    mse_total = 0.0

    with torch.no_grad():
        for images, labels in test_loader:
            images = images.to(device)
            labels = labels.to(device)

            # Get predictions and reconstructions
            z, outputs = classifier(images)
            reconstructed = decoder(z)

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

            # Calculate MSE
            mse = F.mse_loss(reconstructed, images)
            mse_total += mse.item()

    accuracy = 100 * correct / total
    avg_mse = mse_total / len(test_loader)

    return accuracy, avg_mse

# Calculate metrics for each model
```

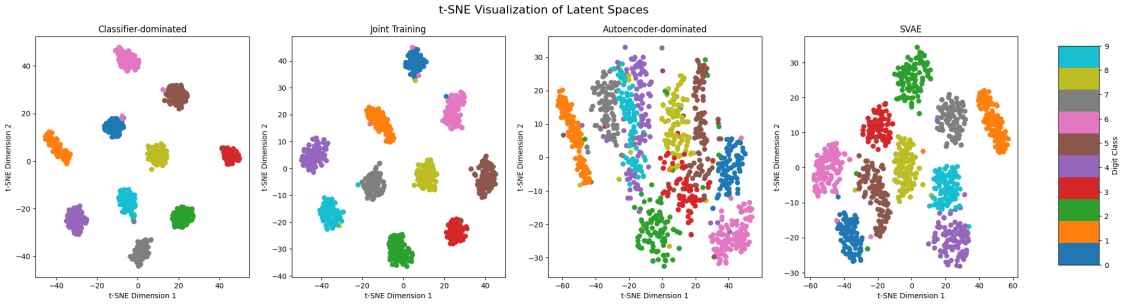
```
model_names = ['Classifier-dominated', 'Joint Training',␣
 ↪'Autoencoder-dominated', 'SVAE']
results = []

for (clf, dec), name in zip(models, model_names):
    accuracy, mse = calculate_metrics(clf, dec, testloader)
    results.append({
        'Model': name,
        'Accuracy (%)': f'{accuracy:.2f}',
        'MSE': f'{mse:.4f}'
    })

# Create and display DataFrame
df = pd.DataFrame(results)
display(df)
```

	Model	Accuracy (%)	MSE
0	Classifier-dominated	99.50	0.0113
1	Joint Training	99.38	0.0033
2	Autoencoder-dominated	95.14	0.0014
3	SVAE	99.14	0.0044

And compare their latent spaces

[69]:
```
# Get sample of training images
random_indices = torch.randint(0, len(testset), (1000,))
images = torch.stack([testset[i][0] for i in random_indices])
labels = torch.tensor([testset[i][1] for i in random_indices])
images = images.to(device)

# Get latent representations from each model
latent_spaces = []
with torch.no_grad():
    for classifier, _ in models:
        classifier.eval()
        latent_reps, _ = classifier(images)
        latent_spaces.append(latent_reps.cpu().numpy())

# Create visualization
fig, axes = plt.subplots(1, 4, figsize=(26, 6))
fig.suptitle('t-SNE Visualization of Latent Spaces', fontsize=16)
model_names = ['Classifier-dominated', 'Joint Training',␣
 ↪'Autoencoder-dominated', 'SVAE']

# Plot each latent space
for i, (latent_reps, name) in enumerate(zip(latent_spaces, model_names)):
    tsne = TSNE(n_components=2, random_state=42)
    latent_2d = tsne.fit_transform(latent_reps)
```

```
    scatter = axes[i].scatter(latent_2d[:, 0], latent_2d[:, 1], c=labels,␣
 ↪cmap='tab10')
    axes[i].set_title(name)
    axes[i].set_xlabel('t-SNE Dimension 1')
    axes[i].set_ylabel('t-SNE Dimension 2')

# Add colorbar
fig.subplots_adjust(right=0.85)
cbar_ax = fig.add_axes([0.88, 0.15, 0.03, 0.7])
fig.colorbar(scatter, cax=cbar_ax, label='Digit Class')

plt.show()
```



[70]:
```
# Get test images
test_images, test_labels = next(iter(testloader))
test_images = test_images.to(device)

# Get reconstructions from each model
reconstructions = []
with torch.no_grad():
    for classifier, decoder in models:
        z, _ = classifier(test_images)
        recon = decoder(z)
        reconstructions.append(recon.cpu().numpy())

# Convert test images to numpy for plotting
test_images = test_images.cpu().numpy()

# Plot original and reconstructed images
fig, axes = plt.subplots(5, 8, figsize=(20, 10))
fig.suptitle('Original vs Reconstructed Images', fontsize=16)

# Plot original images
for i in range(8):
    axes[0,i].imshow(test_images[i,0], cmap='gray')
```

```
    axes[0,i].axis('off')
    if i == 0:
        axes[0,i].set_title('Original', pad=10)

# Plot reconstructions for each model
model_names = ['Classifier-dominated', 'Joint Training',␣
 ↪'Autoencoder-dominated', 'SVAE']
for row, (recon, name) in enumerate(zip(reconstructions, model_names), start=1):
    for i in range(8):
        axes[row,i].imshow(recon[i,0], cmap='gray')
        axes[row,i].axis('off')
        if i == 0:
            axes[row,i].set_title(name, pad=10)

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