assign2

July 14, 2024

[2]:
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
```

This code implements a Variational Autoencoder (VAE) using TensorFlow/Keras. It includes an encoder to compress data into a latent space and a decoder to reconstruct the original data from sampled latent vectors. The VAE allows for efficient representation learning and generation of data.

[11]:
```
class VariationalAutoencoder(tf.keras.Model):
    def __init__(self, input_shape, output_shape, latent_dim=32):
        """
        Initialize the Variational Autoencoder (VAE).

        Args:
        - input_shape: Tuple, shape of the input data (height, width, channels).
        - output_shape: Tuple, shape of the output data (height, width,␣
 ↪channels).
        - latent_dim: Integer, dimensionality of the latent space.
        """
        super(VariationalAutoencoder, self).__init__()
        self.latent_dim = latent_dim

        # Encoder architecture
        self.encoder = tf.keras.Sequential([
            tf.keras.layers.InputLayer(input_shape=input_shape),
            tf.keras.layers.Conv2D(32, 3, strides=2, activation='relu'),
            tf.keras.layers.Conv2D(64, 3, strides=2, activation='relu'),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(latent_dim * 2)
        ])

        # Decoder architecture
        self.decoder = tf.keras.Sequential([
            tf.keras.layers.InputLayer(input_shape=(latent_dim,)),
            tf.keras.layers.Dense(input_shape[0]//4 * input_shape[1]//4 * 64,␣
 ↪activation='relu'),
            tf.keras.layers.Reshape((input_shape[0]//4, input_shape[1]//4, 64)),
            tf.keras.layers.Conv2DTranspose(64, 3, strides=2,␣
↪activation='relu', padding='same'),
            tf.keras.layers.Conv2DTranspose(32, 3, strides=2,␣
↪activation='relu', padding='same'),
            tf.keras.layers.Conv2D(output_shape[-1], 3, activation='sigmoid',␣
↪padding='same')
        ])

    def encode(self, x):
        """
        Encode input data to obtain mean and log-variance parameters of the␣
↪latent distribution.

        Args:
        - x: Tensor, input data.

        Returns:
        - mean: Tensor, mean of the latent distribution.
        - logvar: Tensor, log-variance of the latent distribution.
        """
        mean, logvar = tf.split(self.encoder(x), num_or_size_splits=2, axis=1)
        return mean, logvar

    def reparameterize(self, mean, logvar):
        """
        Reparameterization trick to sample from the latent distribution.

        Args:
        - mean: Tensor, mean of the latent distribution.
        - logvar: Tensor, log-variance of the latent distribution.

        Returns:
        - z: Tensor, sampled latent vector.
        """
        eps = tf.random.normal(shape=mean.shape)
        return eps * tf.exp(logvar * .5) + mean

    def decode(self, z):
        """
        Decode latent vector to reconstruct the input data.

        Args:
        - z: Tensor, latent vector.

        Returns:
        - reconstructed: Tensor, reconstructed data.
        """
        return self.decoder(z)

    def call(self, inputs):
        """
        Forward pass through the VAE.

        Args:
        - inputs: Tensor, input data.

        Returns:
        - reconstructed: Tensor, reconstructed data.
        - mean: Tensor, mean of the latent distribution.
        - logvar: Tensor, log-variance of the latent distribution.
        """
        mean, logvar = self.encode(inputs)
        z = self.reparameterize(mean, logvar)
        return self.decode(z), mean, logvar
```

[16]:

```
class EncoderDecoderVAE(VariationalAutoencoder):
    def __init__(self, input_shape, output_shape, latent_dim=32):
        """
        Initialize the Encoder-Decoder VAE.

        Args:
        - input_shape: Tuple, shape of the input data (height, width, channels).
        - output_shape: Tuple, shape of the output data (height, width,␣
 ↪channels).
        - latent_dim: Integer, dimensionality of the latent space.
        """
        super(EncoderDecoderVAE, self).__init__(input_shape, output_shape,␣
 ↪latent_dim)
        # Customize the encoder architecture if necessary

class DecoderEncoderDecoderVAE(VariationalAutoencoder):
    def __init__(self, input_shape, output_shape, latent_dim=32):
        """
        Initialize the Decoder-Encoder-Decoder VAE.

        Args:
        - input_shape: Tuple, shape of the input data (height, width, channels).
        - output_shape: Tuple, shape of the output data (height, width,␣
 ↪channels).
        - latent_dim: Integer, dimensionality of the latent space.
        """
        super(DecoderEncoderDecoderVAE, self).__init__(input_shape,␣
 ↪output_shape, latent_dim)
        # Customize the decoder architecture if necessary
```

```

class DoubleEncoderDecoderVAE(VariationalAutoencoder):
    def __init__(self, input_shape, output_shape, latent_dim=32):
        """
        Initialize the Double Encoder-Decoder VAE.

        Args:
        - input_shape: Tuple, shape of the input data (height, width, channels).
        - output_shape: Tuple, shape of the output data (height, width,␣
 ↪channels).
        - latent_dim: Integer, dimensionality of the latent space.
        """
        super(DoubleEncoderDecoderVAE, self).__init__(input_shape,␣
 ↪output_shape, latent_dim)
        # Customize both encoder and decoder architectures if necessary
```

[13]:

```
# Function to load and preprocess MNIST data
def load_and_preprocess_data():
    """
    Load and preprocess MNIST dataset.

    Returns:
    - x_train: Numpy array, preprocessed training data.
    - x_test: Numpy array, preprocessed test data.
    """
    (x_train, _), (x_test, _) = mnist.load_data()
    x_train = x_train.astype('float32') / 255.0
    x_test = x_test.astype('float32') / 255.0
    x_train = np.expand_dims(x_train, axis=-1)
    x_test = np.expand_dims(x_test, axis=-1)
    return x_train, x_test

# Function to train and evaluate a VAE model
def train_and_evaluate_vae(model, x_train, x_test, epochs=30, batch_size=128):
    """
    Train and evaluate a VAE model.

    Args:
    - model: Instance of VAE model.
    - x_train: Numpy array, preprocessed training data.
    - x_test: Numpy array, preprocessed test data.
    - epochs: Integer, number of training epochs.
    - batch_size: Integer, batch size for training.

    Returns:
    - test_loss: Float, final test loss.
    - test_losses: List of floats, test losses over epochs.
    """
    optimizer = tf.keras.optimizers.Adam(learning_rate=1e-4)

    @tf.function
    def train_step(x):
        with tf.GradientTape() as tape:
            reconstruction, mean, logvar = model(x)
            reconstruction_loss = tf.reduce_mean(tf.keras.losses.
 ↪binary_crossentropy(x, reconstruction))
            kl_loss = -0.5 * tf.reduce_mean(1 + logvar - tf.square(mean) - tf.
 ↪exp(logvar))
            total_loss = reconstruction_loss + kl_loss
        grads = tape.gradient(total_loss, model.trainable_variables)
        optimizer.apply_gradients(zip(grads, model.trainable_variables))
        return total_loss, reconstruction_loss, kl_loss

    test_losses = []
    for epoch in range(epochs):
        epoch_loss, epoch_reconstruction_loss, epoch_kl_loss = 0, 0, 0
        num_batches = 0

        for batch in tf.data.Dataset.from_tensor_slices(x_train).
 ↪batch(batch_size):
            total_loss, reconstruction_loss, kl_loss = train_step(batch)
            epoch_loss += total_loss
            epoch_reconstruction_loss += reconstruction_loss
            epoch_kl_loss += kl_loss
            num_batches += 1

        epoch_loss /= num_batches
        epoch_reconstruction_loss /= num_batches
        epoch_kl_loss /= num_batches

        test_reconstruction, _, _ = model(x_test)
        test_loss = tf.reduce_mean(tf.keras.losses.binary_crossentropy(x_test,␣
 ↪test_reconstruction))
        test_losses.append(test_loss.numpy())

        print(f'Epoch {epoch + 1}, Train Loss: {epoch_loss:.4f}, Reconstruction␣
 ↪Loss: {epoch_reconstruction_loss:.4f}, KL Loss: {epoch_kl_loss:.4f}, Test␣
 ↪Loss: {test_loss:.4f}')

    return test_losses[-1], test_losses
```

[14]:

```
# Function to plot test losses for all models
def plot_test_losses(model_losses):
    """
    Plot test losses for all VAE models.

    Args:
    - model_losses: Dictionary, mapping model names to lists of test losses.
    """
    plt.figure(figsize=(10, 6))
    for model_name, losses in model_losses.items():
        plt.plot(range(1, len(losses) + 1), losses, label=model_name)
    plt.xlabel('Epoch')
    plt.ylabel('Test Loss')
    plt.title('Test Loss per Epoch for Different VAE Models')
    plt.legend()
    plt.grid(True)
    plt.savefig('vae_models_test_losses.png')
    plt.close()
```

[15]:

```
# Main script to load data, train models, and evaluate
def main():
    x_train, x_test = load_and_preprocess_data()

    models = {
        'VanillaVAE': VariationalAutoencoder(input_shape=(28, 28, 1),␣
 ↪output_shape=(28, 28, 1)),
        'EncoderDecoderVAE': EncoderDecoderVAE(input_shape=(28, 28, 1),␣
 ↪output_shape=(28, 28, 1)),
        'DecoderEncoderDecoderVAE': DecoderEncoderDecoderVAE(input_shape=(28,␣
 ↪28, 1), output_shape=(28, 28, 1)),
        'DoubleEncoderDecoderVAE': DoubleEncoderDecoderVAE(input_shape=(28, 28,␣
 ↪1), output_shape=(28, 28, 1))
    }

    model_losses = {}
    for model_name, model in models.items():
        print(f"\nTraining {model_name}...")
        final_loss, epoch_losses = train_and_evaluate_vae(model, x_train,␣
 ↪x_test)
        model_losses[model_name] = epoch_losses
        print(f'{model_name} Final Test Loss: {final_loss:.4f}')

    plot_test_losses(model_losses)

    # Find the best performing model
    best_model = min(model_losses, key=lambda x: model_losses[x][-1])
    print(f"\nBest performing model: {best_model}")

if __name__ == "__main__":
    main()
```

```
Training VanillaVAE…
Epoch 1, Train Loss: 0.3839, Reconstruction Loss: 0.3815, KL Loss: 0.0024, Test
Loss: 0.2744
Epoch 2, Train Loss: 0.2704, Reconstruction Loss: 0.2687, KL Loss: 0.0018, Test
Loss: 0.2662
Epoch 3, Train Loss: 0.2673, Reconstruction Loss: 0.2656, KL Loss: 0.0018, Test
Loss: 0.2644
Epoch 4, Train Loss: 0.2660, Reconstruction Loss: 0.2643, KL Loss: 0.0018, Test
Loss: 0.2633
Epoch 5, Train Loss: 0.2651, Reconstruction Loss: 0.2634, KL Loss: 0.0018, Test
Loss: 0.2625
Epoch 6, Train Loss: 0.2646, Reconstruction Loss: 0.2628, KL Loss: 0.0018, Test
Loss: 0.2623
Epoch 7, Train Loss: 0.2643, Reconstruction Loss: 0.2625, KL Loss: 0.0018, Test
Loss: 0.2619
Epoch 8, Train Loss: 0.2641, Reconstruction Loss: 0.2622, KL Loss: 0.0019, Test
Loss: 0.2616
Epoch 9, Train Loss: 0.2640, Reconstruction Loss: 0.2621, KL Loss: 0.0019, Test
Loss: 0.2616
Epoch 10, Train Loss: 0.2637, Reconstruction Loss: 0.2617, KL Loss: 0.0020, Test
Loss: 0.2613
Epoch 11, Train Loss: 0.2637, Reconstruction Loss: 0.2617, KL Loss: 0.0020, Test
Loss: 0.2612
Epoch 12, Train Loss: 0.2636, Reconstruction Loss: 0.2616, KL Loss: 0.0020, Test
Loss: 0.2608
Epoch 13, Train Loss: 0.2635, Reconstruction Loss: 0.2614, KL Loss: 0.0021, Test
Loss: 0.2610
Epoch 14, Train Loss: 0.2635, Reconstruction Loss: 0.2614, KL Loss: 0.0021, Test
Loss: 0.2608
Epoch 15, Train Loss: 0.2633, Reconstruction Loss: 0.2612, KL Loss: 0.0021, Test
Loss: 0.2609
Epoch 16, Train Loss: 0.2634, Reconstruction Loss: 0.2613, KL Loss: 0.0021, Test
Loss: 0.2606
Epoch 17, Train Loss: 0.2633, Reconstruction Loss: 0.2611, KL Loss: 0.0021, Test
Loss: 0.2607
Epoch 18, Train Loss: 0.2633, Reconstruction Loss: 0.2611, KL Loss: 0.0021, Test
Loss: 0.2609
Epoch 19, Train Loss: 0.2633, Reconstruction Loss: 0.2611, KL Loss: 0.0021, Test
Loss: 0.2605
Epoch 20, Train Loss: 0.2631, Reconstruction Loss: 0.2610, KL Loss: 0.0022, Test
Loss: 0.2604
Epoch 21, Train Loss: 0.2631, Reconstruction Loss: 0.2609, KL Loss: 0.0022, Test
Loss: 0.2605
Epoch 22, Train Loss: 0.2631, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2604
Epoch 23, Train Loss: 0.2631, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2605
```

```
Epoch 24, Train Loss: 0.2631, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2605
Epoch 25, Train Loss: 0.2631, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2604
Epoch 26, Train Loss: 0.2630, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2603
Epoch 27, Train Loss: 0.2631, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2604
Epoch 28, Train Loss: 0.2631, Reconstruction Loss: 0.2609, KL Loss: 0.0022, Test
Loss: 0.2601
Epoch 29, Train Loss: 0.2630, Reconstruction Loss: 0.2607, KL Loss: 0.0023, Test
Loss: 0.2601
Epoch 30, Train Loss: 0.2629, Reconstruction Loss: 0.2606, KL Loss: 0.0023, Test
Loss: 0.2601
VanillaVAE Final Test Loss: 0.2601

Training EncoderDecoderVAE…
Epoch 1, Train Loss: 0.3875, Reconstruction Loss: 0.3856, KL Loss: 0.0019, Test
Loss: 0.2739
Epoch 2, Train Loss: 0.2702, Reconstruction Loss: 0.2687, KL Loss: 0.0016, Test
Loss: 0.2664
Epoch 3, Train Loss: 0.2672, Reconstruction Loss: 0.2655, KL Loss: 0.0017, Test
Loss: 0.2645
Epoch 4, Train Loss: 0.2659, Reconstruction Loss: 0.2642, KL Loss: 0.0017, Test
Loss: 0.2635
Epoch 5, Train Loss: 0.2650, Reconstruction Loss: 0.2633, KL Loss: 0.0018, Test
Loss: 0.2626
Epoch 6, Train Loss: 0.2647, Reconstruction Loss: 0.2629, KL Loss: 0.0018, Test
Loss: 0.2622
Epoch 7, Train Loss: 0.2644, Reconstruction Loss: 0.2626, KL Loss: 0.0018, Test
Loss: 0.2620
Epoch 8, Train Loss: 0.2640, Reconstruction Loss: 0.2623, KL Loss: 0.0018, Test
Loss: 0.2619
Epoch 9, Train Loss: 0.2639, Reconstruction Loss: 0.2620, KL Loss: 0.0019, Test
Loss: 0.2614
Epoch 10, Train Loss: 0.2638, Reconstruction Loss: 0.2620, KL Loss: 0.0019, Test
Loss: 0.2614
Epoch 11, Train Loss: 0.2637, Reconstruction Loss: 0.2618, KL Loss: 0.0019, Test
Loss: 0.2613
Epoch 12, Train Loss: 0.2636, Reconstruction Loss: 0.2616, KL Loss: 0.0019, Test
Loss: 0.2612
Epoch 13, Train Loss: 0.2635, Reconstruction Loss: 0.2615, KL Loss: 0.0020, Test
Loss: 0.2611
Epoch 14, Train Loss: 0.2633, Reconstruction Loss: 0.2613, KL Loss: 0.0020, Test
Loss: 0.2608
Epoch 15, Train Loss: 0.2634, Reconstruction Loss: 0.2612, KL Loss: 0.0021, Test
Loss: 0.2610
Epoch 16, Train Loss: 0.2633, Reconstruction Loss: 0.2612, KL Loss: 0.0021, Test
```

```
Loss: 0.2609
Epoch 17, Train Loss: 0.2633, Reconstruction Loss: 0.2612, KL Loss: 0.0021, Test
Loss: 0.2608
Epoch 18, Train Loss: 0.2633, Reconstruction Loss: 0.2612, KL Loss: 0.0021, Test
Loss: 0.2606
Epoch 19, Train Loss: 0.2632, Reconstruction Loss: 0.2611, KL Loss: 0.0021, Test
Loss: 0.2608
Epoch 20, Train Loss: 0.2631, Reconstruction Loss: 0.2609, KL Loss: 0.0022, Test
Loss: 0.2606
Epoch 21, Train Loss: 0.2631, Reconstruction Loss: 0.2609, KL Loss: 0.0023, Test
Loss: 0.2603
Epoch 22, Train Loss: 0.2631, Reconstruction Loss: 0.2609, KL Loss: 0.0022, Test
Loss: 0.2605
Epoch 23, Train Loss: 0.2631, Reconstruction Loss: 0.2609, KL Loss: 0.0022, Test
Loss: 0.2606
Epoch 24, Train Loss: 0.2631, Reconstruction Loss: 0.2608, KL Loss: 0.0022, Test
Loss: 0.2603
Epoch 25, Train Loss: 0.2631, Reconstruction Loss: 0.2609, KL Loss: 0.0022, Test
Loss: 0.2603
Epoch 26, Train Loss: 0.2631, Reconstruction Loss: 0.2609, KL Loss: 0.0022, Test
Loss: 0.2603
Epoch 27, Train Loss: 0.2630, Reconstruction Loss: 0.2608, KL Loss: 0.0022, Test
Loss: 0.2604
Epoch 28, Train Loss: 0.2630, Reconstruction Loss: 0.2608, KL Loss: 0.0022, Test
Loss: 0.2605
Epoch 29, Train Loss: 0.2629, Reconstruction Loss: 0.2606, KL Loss: 0.0023, Test
Loss: 0.2602
Epoch 30, Train Loss: 0.2630, Reconstruction Loss: 0.2607, KL Loss: 0.0023, Test
Loss: 0.2602
EncoderDecoderVAE Final Test Loss: 0.2602

Training DecoderEncoderDecoderVAE…
Epoch 1, Train Loss: 0.3847, Reconstruction Loss: 0.3824, KL Loss: 0.0023, Test
Loss: 0.2739
Epoch 2, Train Loss: 0.2703, Reconstruction Loss: 0.2687, KL Loss: 0.0016, Test
Loss: 0.2660
Epoch 3, Train Loss: 0.2673, Reconstruction Loss: 0.2657, KL Loss: 0.0016, Test
Loss: 0.2647
Epoch 4, Train Loss: 0.2662, Reconstruction Loss: 0.2644, KL Loss: 0.0017, Test
Loss: 0.2635
Epoch 5, Train Loss: 0.2652, Reconstruction Loss: 0.2634, KL Loss: 0.0018, Test
Loss: 0.2628
Epoch 6, Train Loss: 0.2647, Reconstruction Loss: 0.2628, KL Loss: 0.0018, Test
Loss: 0.2622
Epoch 7, Train Loss: 0.2642, Reconstruction Loss: 0.2623, KL Loss: 0.0019, Test
Loss: 0.2621
Epoch 8, Train Loss: 0.2640, Reconstruction Loss: 0.2620, KL Loss: 0.0020, Test
Loss: 0.2615
```

```
Epoch 9, Train Loss: 0.2639, Reconstruction Loss: 0.2618, KL Loss: 0.0020, Test
Loss: 0.2614
Epoch 10, Train Loss: 0.2639, Reconstruction Loss: 0.2619, KL Loss: 0.0020, Test
Loss: 0.2612
Epoch 11, Train Loss: 0.2637, Reconstruction Loss: 0.2617, KL Loss: 0.0020, Test
Loss: 0.2611
Epoch 12, Train Loss: 0.2636, Reconstruction Loss: 0.2616, KL Loss: 0.0020, Test
Loss: 0.2612
Epoch 13, Train Loss: 0.2635, Reconstruction Loss: 0.2615, KL Loss: 0.0020, Test
Loss: 0.2611
Epoch 14, Train Loss: 0.2634, Reconstruction Loss: 0.2613, KL Loss: 0.0021, Test
Loss: 0.2610
Epoch 15, Train Loss: 0.2634, Reconstruction Loss: 0.2612, KL Loss: 0.0021, Test
Loss: 0.2609
Epoch 16, Train Loss: 0.2633, Reconstruction Loss: 0.2611, KL Loss: 0.0021, Test
Loss: 0.2607
Epoch 17, Train Loss: 0.2633, Reconstruction Loss: 0.2611, KL Loss: 0.0022, Test
Loss: 0.2607
Epoch 18, Train Loss: 0.2632, Reconstruction Loss: 0.2609, KL Loss: 0.0023, Test
Loss: 0.2605
Epoch 19, Train Loss: 0.2632, Reconstruction Loss: 0.2609, KL Loss: 0.0023, Test
Loss: 0.2605
Epoch 20, Train Loss: 0.2632, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2608
Epoch 21, Train Loss: 0.2632, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2604
Epoch 22, Train Loss: 0.2631, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2603
Epoch 23, Train Loss: 0.2631, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2604
Epoch 24, Train Loss: 0.2631, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2602
Epoch 25, Train Loss: 0.2631, Reconstruction Loss: 0.2607, KL Loss: 0.0024, Test
Loss: 0.2603
Epoch 26, Train Loss: 0.2631, Reconstruction Loss: 0.2608, KL Loss: 0.0023, Test
Loss: 0.2603
Epoch 27, Train Loss: 0.2629, Reconstruction Loss: 0.2604, KL Loss: 0.0025, Test
Loss: 0.2598
Epoch 28, Train Loss: 0.2630, Reconstruction Loss: 0.2604, KL Loss: 0.0026, Test
Loss: 0.2600
Epoch 29, Train Loss: 0.2629, Reconstruction Loss: 0.2603, KL Loss: 0.0026, Test
Loss: 0.2600
Epoch 30, Train Loss: 0.2631, Reconstruction Loss: 0.2605, KL Loss: 0.0026, Test
Loss: 0.2600
DecoderEncoderDecoderVAE Final Test Loss: 0.2600

Training DoubleEncoderDecoderVAE…
Epoch 1, Train Loss: 0.3800, Reconstruction Loss: 0.3779, KL Loss: 0.0022, Test
```

```
Loss: 0.2727
Epoch 2, Train Loss: 0.2700, Reconstruction Loss: 0.2683, KL Loss: 0.0017, Test
Loss: 0.2660
Epoch 3, Train Loss: 0.2672, Reconstruction Loss: 0.2654, KL Loss: 0.0017, Test
Loss: 0.2646
Epoch 4, Train Loss: 0.2658, Reconstruction Loss: 0.2640, KL Loss: 0.0019, Test
Loss: 0.2632
Epoch 5, Train Loss: 0.2650, Reconstruction Loss: 0.2631, KL Loss: 0.0019, Test
Loss: 0.2624
Epoch 6, Train Loss: 0.2646, Reconstruction Loss: 0.2628, KL Loss: 0.0019, Test
Loss: 0.2623
Epoch 7, Train Loss: 0.2643, Reconstruction Loss: 0.2623, KL Loss: 0.0020, Test
Loss: 0.2617
Epoch 8, Train Loss: 0.2640, Reconstruction Loss: 0.2619, KL Loss: 0.0021, Test
Loss: 0.2615
Epoch 9, Train Loss: 0.2638, Reconstruction Loss: 0.2617, KL Loss: 0.0022, Test
Loss: 0.2612
Epoch 10, Train Loss: 0.2637, Reconstruction Loss: 0.2616, KL Loss: 0.0022, Test
Loss: 0.2613
Epoch 11, Train Loss: 0.2636, Reconstruction Loss: 0.2614, KL Loss: 0.0022, Test
Loss: 0.2611
Epoch 12, Train Loss: 0.2636, Reconstruction Loss: 0.2613, KL Loss: 0.0022, Test
Loss: 0.2611
Epoch 13, Train Loss: 0.2635, Reconstruction Loss: 0.2613, KL Loss: 0.0022, Test
Loss: 0.2609
Epoch 14, Train Loss: 0.2635, Reconstruction Loss: 0.2612, KL Loss: 0.0023, Test
Loss: 0.2607
Epoch 15, Train Loss: 0.2634, Reconstruction Loss: 0.2611, KL Loss: 0.0023, Test
Loss: 0.2607
Epoch 16, Train Loss: 0.2633, Reconstruction Loss: 0.2609, KL Loss: 0.0024, Test
Loss: 0.2603
Epoch 17, Train Loss: 0.2632, Reconstruction Loss: 0.2608, KL Loss: 0.0024, Test
Loss: 0.2604
Epoch 18, Train Loss: 0.2633, Reconstruction Loss: 0.2609, KL Loss: 0.0024, Test
Loss: 0.2604
Epoch 19, Train Loss: 0.2632, Reconstruction Loss: 0.2608, KL Loss: 0.0024, Test
Loss: 0.2606
Epoch 20, Train Loss: 0.2632, Reconstruction Loss: 0.2608, KL Loss: 0.0024, Test
Loss: 0.2602
Epoch 21, Train Loss: 0.2632, Reconstruction Loss: 0.2608, KL Loss: 0.0024, Test
Loss: 0.2604
Epoch 22, Train Loss: 0.2631, Reconstruction Loss: 0.2607, KL Loss: 0.0024, Test
Loss: 0.2604
Epoch 23, Train Loss: 0.2632, Reconstruction Loss: 0.2607, KL Loss: 0.0025, Test
Loss: 0.2601
Epoch 24, Train Loss: 0.2631, Reconstruction Loss: 0.2606, KL Loss: 0.0025, Test
Loss: 0.2601
Epoch 25, Train Loss: 0.2630, Reconstruction Loss: 0.2604, KL Loss: 0.0026, Test
```

```
Loss: 0.2598
Epoch 26, Train Loss: 0.2630, Reconstruction Loss: 0.2604, KL Loss: 0.0027, Test
Loss: 0.2599
Epoch 27, Train Loss: 0.2630, Reconstruction Loss: 0.2603, KL Loss: 0.0027, Test
Loss: 0.2598
Epoch 28, Train Loss: 0.2630, Reconstruction Loss: 0.2603, KL Loss: 0.0027, Test
Loss: 0.2599
Epoch 29, Train Loss: 0.2629, Reconstruction Loss: 0.2601, KL Loss: 0.0028, Test
Loss: 0.2599
Epoch 30, Train Loss: 0.2629, Reconstruction Loss: 0.2601, KL Loss: 0.0029, Test
Loss: 0.2593
DoubleEncoderDecoderVAE Final Test Loss: 0.2593

Best performing model: DoubleEncoderDecoderVAE
```