

Experiment No.2 Autoencoders and Variational Autoencoders (VAE)

1. Build a simple AE model for Dimensionality Reduction and Denoising.
2. Generate realistic faces or interpolate between facial features for creative applications in entertainment and design using VAE.

Name: Nabil Ansari

Batch: GAA Lab 3

PRN/Roll no.: 202302040004

Colab Link: https://colab.research.google.com/drive/1GNVV_e3Z0BcWzZVYLO0iFOdo9Ge-bAVH?usp=sharing

1. Build a simple AE model for Dimensionality Reduction and Denoising.

Understanding Autoencoders

1. What an Autoencoder Is

An **Autoencoder** is a neural network trained to copy its input to its output through a compressed intermediate representation.

It has two main parts:

Part	Function
Encoder	Compresses the input into a smaller latent vector (dimensionality reduction)
Decoder	Reconstructs the original input from the latent vector

2. Dimensionality Reduction

- The encoder transforms an input image x into a latent vector z of much lower dimension than the original.
- This forces the model to learn an **efficient representation** of the input.
- You can then use this latent vector z for:
 - Feature extraction
 - Visualization (with PCA/t-SNE)
 - Feeding into other models

Example: An MNIST image of shape $28 \times 28 \times 1 = 784$ pixels → encoded into a 64-dimensional vector.

3. Denoising

- During training you feed the **noisy version** of the input but ask the model to output the **clean version**.
- This encourages the latent representation to capture only the **signal** and ignore the noise.
- At inference time you can pass in noisy images and the AE outputs denoised versions.

Setup

Install necessary libraries and import dependencies.

```
In [ ]: !pip install tensorflow tensorflow-datasets
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.12/dist-packages (2.19.0)
Requirement already satisfied: tensorflow-datasets in /usr/local/lib/python3.12/dist-packages (4.9.9)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.9.23)
Requirement already satisfied: gast!=0.5.0,!0.5.1,!0.5.2,>0.2.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.0)
Requirement already satisfied: protobuf!=4.21.0,!4.21.1,!4.21.2,!4.21.3,!4.21.4,!4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (5.29.5)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.32.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.2.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (4.15.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.0.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.76.0)
Requirement already satisfied: tensorboard~=2.19.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.19.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.10.0)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.15.1)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.5.3)
Requirement already satisfied: array_record>=0.5.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow-datasets) (0.8.2)
Requirement already satisfied: dm-tree in /usr/local/lib/python3.12/dist-packages (from tensorflow-datasets) (0.1.9)
Requirement already satisfied: etils>=1.9.1 in /usr/local/lib/python3.12/dist-packages (from etils[edc,epn,epath,epy,etree]>=1.9.1; python_version >= "3.11"->tensorflow-datasets) (1.13.0)
Requirement already satisfied: immutabledict in /usr/local/lib/python3.12/dist-packages (from tensorflow-datasets) (4.2.2)
```

Requirement already satisfied: promise in /usr/local/lib/python3.12/dist-packages (from tensorflow-datasets) (2.3)

Requirement already satisfied: psutil in /usr/local/lib/python3.12/dist-packages (from tensorflow-datasets) (5.9.5)

Requirement already satisfied: pyarrow in /usr/local/lib/python3.12/dist-packages (from tensorflow-datasets) (18.1.0)

Requirement already satisfied: simple_parsing in /usr/local/lib/python3.12/dist-packages (from tensorflow-datasets) (0.1.7)

Requirement already satisfied: tensorflow-metadata in /usr/local/lib/python3.12/dist-packages (from tensorflow-datasets) (1.17.2)

Requirement already satisfied: toml in /usr/local/lib/python3.12/dist-packages (from tensorflow-datasets) (0.10.2)

Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from tensorflow-datasets) (4.67.1)

Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from astunparse>=1.6.0->tensorflow) (0.45.1)

Requirement already satisfied: einops in /usr/local/lib/python3.12/dist-packages (from etils[edc,epn,epath,epy,etree]>=1.9.1; python_version >= "3.11"->tensorflow-datasets) (0.8.1)

Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from etils[edc,epn,epath,epy,etree]>=1.9.1; python_version >= "3.11"->tensorflow-datasets) (2025.3.0)

Requirement already satisfied: importlib_resources in /usr/local/lib/python3.12/dist-packages (from etils[edc,epn,epath,epy,etree]>=1.9.1; python_version >= "3.11"->tensorflow-datasets) (6.5.2)

Requirement already satisfied: zipp in /usr/local/lib/python3.12/dist-packages (from etils[edc,epn,epath,epy,etree]>=1.9.1; python_version >= "3.11"->tensorflow-datasets) (3.23.0)

Requirement already satisfied: rich in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)

Requirement already satisfied: namex in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0.1.0)

Requirement already satisfied: optree in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0.17.0)

Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.4.4)

Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.11)

Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.5.0)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (2025.10.5)

Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.12/dist-packages (from tensorboard~2.19.0->tensorflow) (3.10)

Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from tensorboard~2.19.0->tensorflow) (0.7.2)

Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from tensorboard~2.19.0->tensorflow) (3.1.3)

Requirement already satisfied: attrs>=18.2.0 in /usr/local/lib/python3.12/dist-packages (from dm-tree->tensorflow-datasets) (25.4.0)

Requirement already satisfied: docstring-parser<1.0,>=0.15 in /usr/local/lib/python3.12/dist-packages (from simple_parsing->tensorflow-datasets) (0.17.0)

Requirement already satisfied: googleapis-common-protos<2,>=1.56.4 in /usr/local/lib/python3.12/dist-packages (from tensorflow-metadata->tensorflow-datasets) (1.72.0)

Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from werkzeug>=1.0.1->tensorboard~2.19.0->tensorflow) (3.0.3)

```
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3.5.0->tensorflow) (4.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3.5.0->tensorflow) (2.19.2)
Requirement already satisfied: mdurl~0.1 in /usr/local/lib/python3.12/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.2)
```

```
In [ ]: import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt

# Load MNIST
(x_train, _), (x_test, _) = tf.keras.datasets.mnist.load_data()
x_train = x_train.astype("float32") / 255.
x_test = x_test.astype("float32") / 255.
x_train = np.expand_dims(x_train, -1) # (batch, 28, 28, 1)
x_test = np.expand_dims(x_test, -1)

# Add random noise for denoising AE
noise_factor = 0.5
x_train_noisy = np.clip(x_train + noise_factor * np.random.normal(size=x_train.shape), 0., 1.)
x_test_noisy = np.clip(x_test + noise_factor * np.random.normal(size=x_test.shape), 0., 1.)

# Build simple convolutional AE
latent_dim = 64

encoder_input = layers.Input(shape=(28, 28, 1))
x = layers.Conv2D(32, (3,3), activation='relu', padding='same')(encoder_input)
x = layers.MaxPooling2D((2,2), padding='same')(x)
x = layers.Conv2D(64, (3,3), activation='relu', padding='same')(x)
x = layers.MaxPooling2D((2,2), padding='same')(x)
x = layers.Flatten()(x)
latent = layers.Dense(latent_dim, name="latent")(x)

# Decoder
x = layers.Dense(7*7*64, activation='relu')(latent)
x = layers.Reshape((7,7,64))(x)
x = layers.Conv2DTranspose(64, (3,3), strides=2, activation='relu', padding='same')
x = layers.Conv2DTranspose(32, (3,3), strides=2, activation='relu', padding='same')
decoder_output = layers.Conv2DTranspose(1, (3,3), activation='sigmoid', padding='same')

autoencoder = models.Model(encoder_input, decoder_output)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
autoencoder.summary()

# Train
autoencoder.fit(x_train_noisy, x_train,
                epochs=10,
                batch_size=128,
                shuffle=True,
                validation_data=(x_test_noisy, x_test))

# Get Latent representations (dimensionality reduction)
encoder = models.Model(encoder_input, latent)
latent_repr = encoder.predict(x_test_noisy)
```

```

print("Latent shape:", latent_repr.shape) # (num_samples, Latent_dim)

# Visualize denoising
decoded_imgs = autoencoder.predict(x_test_noisy)

n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    # Noisy input
    ax = plt.subplot(3, n, i + 1)
    plt.imshow(x_test_noisy[i].reshape(28,28), cmap='gray')
    plt.title("Noisy")
    plt.axis('off')

    # Denoised output
    ax = plt.subplot(3, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28,28), cmap='gray')
    plt.title("Denoised")
    plt.axis('off')

    # Original
    ax = plt.subplot(3, n, i + 1 + 2*n)
    plt.imshow(x_test[i].reshape(28,28), cmap='gray')
    plt.title("Original")
    plt.axis('off')
plt.show()

```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>
11490434/11490434 ————— 0s 0us/step
Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 28, 28, 1)	0
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 14, 14, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 64)	0
flatten (Flatten)	(None, 3136)	0
latent (Dense)	(None, 64)	200,768
dense (Dense)	(None, 3136)	203,840
reshape (Reshape)	(None, 7, 7, 64)	0
conv2d_transpose (Conv2DTranspose)	(None, 14, 14, 64)	36,928
conv2d_transpose_1 (Conv2DTranspose)	(None, 28, 28, 32)	18,464
conv2d_transpose_2 (Conv2DTranspose)	(None, 28, 28, 1)	289

Total params: 479,105 (1.83 MB)

Trainable params: 479,105 (1.83 MB)

Non-trainable params: 0 (0.00 B)

```

Epoch 1/10
469/469 13s 16ms/step - loss: 0.2816 - val_loss: 0.1200
Epoch 2/10
469/469 4s 8ms/step - loss: 0.1148 - val_loss: 0.1036
Epoch 3/10
469/469 4s 8ms/step - loss: 0.1027 - val_loss: 0.0994
Epoch 4/10
469/469 4s 8ms/step - loss: 0.0986 - val_loss: 0.0972
Epoch 5/10
469/469 4s 8ms/step - loss: 0.0962 - val_loss: 0.0957
Epoch 6/10
469/469 4s 8ms/step - loss: 0.0942 - val_loss: 0.0952
Epoch 7/10
469/469 4s 9ms/step - loss: 0.0931 - val_loss: 0.0944
Epoch 8/10
469/469 4s 8ms/step - loss: 0.0922 - val_loss: 0.0934
Epoch 9/10
469/469 4s 8ms/step - loss: 0.0910 - val_loss: 0.0931
Epoch 10/10
469/469 4s 9ms/step - loss: 0.0902 - val_loss: 0.0928
313/313 1s 2ms/step
Latent shape: (10000, 64)
313/313 1s 3ms/step


```

2. Generate realistic faces or interpolate between facial features for creative applications in entertainment and design using VAE

This demonstrates how to build and train convolutional autoencoders (AE) and variational autoencoders (VAE) using TensorFlow and Keras for image denoising and face generation.

```
In [17]: import os
import math
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow_datasets as tfds
import cv2
```

```
print("TensorFlow version:", tf.__version__)
```

```
TensorFlow version: 2.19.0
```

Configuration

Define parameters for image size, batch size, latent dimension, and training epochs.

```
In [ ]: IMAGE_SIZE = 64          # resize images to 64x64 (balanced speed-quality). Use 128 instead for better quality
BATCH_SIZE = 128
LATENT_DIM = 128          # Latent space size
EPOCHS_AE = 100           # AE epochs
EPOCHS_VAE = 100           # VAE epochs
AUTOTUNE = tf.data.AUTOTUNE
```

Load and Prepare Dataset

Load the Labeled Faces in the Wild (LFW) dataset using `tensorflow_datasets`.

Preprocess the images by resizing and normalizing them.

```
In [ ]: def preprocess_example(example):
    # example is a dict from tfds for celeb_a that has 'image' key
    img = example['image']
    img = tf.image.resize(img, [IMAGE_SIZE, IMAGE_SIZE])
    img = tf.cast(img, tf.float32) / 127.5 - 1.0 # normalize to [-1, +1]
    return img

print("Loading dataset (this may download ~1-2GB). If you want smaller dataset, change dataset_name to 'lfw_people'")

# If CelebA isn't desirable, you can switch to 'lfw_people' by changing dataset_name
try:
    dataset_name = 'lfw'
    ds, ds_info = tfds.load(dataset_name, split='train', shuffle_files=True, with_info=True)
    total_examples = ds_info.splits['train'].num_examples
    print(f"Loaded {dataset_name} with {total_examples} images")
except Exception as e:
    print("Couldn't load CelebA via tfds. Falling back to lfw_people (smaller). Error was: " + str(e))
    dataset_name = 'lfw_people'
    ds, ds_info = tfds.load(dataset_name, split='train', shuffle_files=True, with_info=True)
    total_examples = ds_info.splits['train'].num_examples
    print(f"Loaded {dataset_name} with {total_examples} images")

# Map and batch
ds = ds.map(preprocess_example, num_parallel_calls=AUTOTUNE)
# cache for speed (if memory available)
# ds = ds.cache()
# shuffle and batch
ds = ds.shuffle(2048).batch(BATCH_SIZE).prefetch(AUTOTUNE)

# For quick prototyping we also prepare a smaller test sample
sample_iter = ds.take(1)
for batch in sample_iter:
```

```
sample_images = batch[0:16] if isinstance(batch, tf.Tensor) else batch[:16]
break
```

```
Loading dataset (this may download ~1-2GB). If you want smaller dataset, change `with_h_info` and/or dataset name.
Downloading and preparing dataset Unknown size (download: Unknown size, generated: Unknown size, total: Unknown size) to /root/tensorflow_datasets/lfw/0.1.1...
Dl Completed...: 0 url [00:00, ? url/s]
Dl Size...: 0 MiB [00:00, ? MiB/s]
Extraction completed...: 0 file [00:00, ? file/s]
Generating splits...: 0% | 0/1 [00:00<?, ? splits/s]
Generating train examples...: 0 examples [00:00, ? examples/s]
Shuffling /root/tensorflow_datasets/lfw/incomplete.E02PC0_0.1.1/lfw-train.tfrecord
*...: 0% | 0/132...
Dataset lfw downloaded and prepared to /root/tensorflow_datasets/lfw/0.1.1. Subsequent calls will reuse this data.
Loaded lfw with 13233 images
```

Helper to show images

```
In [ ]: def show_images(imgs, ncols=8, title=None):
    imgs = (imgs + 1.0) * 127.5 # [-1,1] -> [0,255]
    # Check if imgs is a TensorFlow tensor before calling .numpy()
    if isinstance(imgs, tf.Tensor):
        imgs = imgs.numpy().astype(np.uint8)
    else:
        imgs = imgs.astype(np.uint8) # Assume it's already a NumPy array
    n = imgs.shape[0]
    nrows = math.ceil(n / ncols)
    plt.figure(figsize=(ncols * 1.6, nrows * 1.6))
    for i in range(n):
        plt.subplot(nrows, ncols, i + 1)
        plt.imshow(imgs[i])
        plt.axis('off')
    if title:
        plt.suptitle(title)
    plt.show()

# show a few real images
for batch in ds.take(1):
    real_sample = batch[:16]
    break

print('Real sample:')
show_images(real_sample)
```

Real sample:



Convolutional Autoencoder (denoising + dimensionality reduction)

```
In [ ]: # Encoder
encoder_inputs = layers.Input(shape=(IMAGE_SIZE, IMAGE_SIZE, 3))
# Add Gaussian noise for denoising ability at train-time via a separate input or us
x = encoder_inputs
x = layers.Normalization()(x) # optional - we already normalized manually, keep id

x = layers.Conv2D(32, 3, strides=2, padding='same', activation='relu')(x) # 32x32
x = layers.Conv2D(64, 3, strides=2, padding='same', activation='relu')(x) # 16x16
x = layers.Conv2D(128, 3, strides=2, padding='same', activation='relu')(x) # 8x8
x = layers.Conv2D(256, 3, strides=2, padding='same', activation='relu')(x) # 4x4
shape_before_flatten = tf.keras.backend.int_shape(x)[1:]
x = layers.Flatten()(x)
latent = layers.Dense(LATENT_DIM, name='latent_vector')(x)
encoder = keras.Model(encoder_inputs, latent, name='encoder')
encoder.summary()

# Decoder
latent_inputs = layers.Input(shape=(LATENT_DIM,))
x = layers.Dense(np.prod(shape_before_flatten), activation='relu')(latent_inputs)
x = layers.Reshape(shape_before_flatten)(x)
x = layers.Conv2DTranspose(256, 3, strides=2, padding='same', activation='relu')(x)
x = layers.Conv2DTranspose(128, 3, strides=2, padding='same', activation='relu')(x)
x = layers.Conv2DTranspose(64, 3, strides=2, padding='same', activation='relu')(x)
x = layers.Conv2DTranspose(32, 3, strides=2, padding='same', activation='relu')(x)
# final layer, tanh to match normalized [-1,1]
decoder_outputs = layers.Conv2D(3, 3, activation='tanh', padding='same')(x)
decoder = keras.Model(latent_inputs, decoder_outputs, name='decoder')
decoder.summary()

# Autoencoder = encoder + decoder
ae_inputs = encoder_inputs
ae_latent = encoder(ae_inputs)
ae_outputs = decoder(ae_latent)
ae = keras.Model(ae_inputs, ae_outputs, name='autoencoder')

# Loss and compile
ae.compile(optimizer=keras.optimizers.Adam(1e-4), loss='mse')
```

Model: "encoder"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 64, 64, 3)	0
normalization (Normalization)	(None, 64, 64, 3)	7
conv2d_2 (Conv2D)	(None, 32, 32, 32)	896
conv2d_3 (Conv2D)	(None, 16, 16, 64)	18,496
conv2d_4 (Conv2D)	(None, 8, 8, 128)	73,856
conv2d_5 (Conv2D)	(None, 4, 4, 256)	295,168
flatten_1 (Flatten)	(None, 4096)	0
latent_vector (Dense)	(None, 128)	524,416

Total params: 912,839 (3.48 MB)

Trainable params: 912,832 (3.48 MB)

Non-trainable params: 7 (32.00 B)

Model: "decoder"

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 128)	0
dense_1 (Dense)	(None, 4096)	528,384
reshape_1 (Reshape)	(None, 4, 4, 256)	0
conv2d_transpose_3 (Conv2DTranspose)	(None, 8, 8, 256)	590,080
conv2d_transpose_4 (Conv2DTranspose)	(None, 16, 16, 128)	295,040
conv2d_transpose_5 (Conv2DTranspose)	(None, 32, 32, 64)	73,792
conv2d_transpose_6 (Conv2DTranspose)	(None, 64, 64, 32)	18,464
conv2d_6 (Conv2D)	(None, 64, 64, 3)	867

Total params: 1,506,627 (5.75 MB)

Trainable params: 1,506,627 (5.75 MB)

Non-trainable params: 0 (0.00 B)

Prepare noisy dataset for denoising

```
In [ ]: NOISE_FACTOR = 0.2

def add_noise(images):
    noise = tf.random.normal(shape=tf.shape(images), mean=0.0, stddev=NOISE_FACTOR)
    noisy = images + noise
    noisy = tf.clip_by_value(noisy, -1.0, 1.0)
    return noisy

# Create dataset pairs: (noisy_image, clean_image)
paired_ds = ds.map(lambda x: (add_noise(x), x), num_parallel_calls=AUTOTUNE)
paired_ds = paired_ds.prefetch(AUTOTUNE)
```

Train Autoencoder (Denoising) & Visualize AE denoising results

```
In [ ]: checkpoint_dir = '/content/checkpoints_ae'
os.makedirs(checkpoint_dir, exist_ok=True)

callbacks = [
    keras.callbacks.ModelCheckpoint(os.path.join(checkpoint_dir, 'ae_best.h5'), save_best_only=True),
    keras.callbacks.ReduceLROnPlateau(monitor='loss', factor=0.5, patience=3, verbose=1)
]

print('Training AE (denoising) ...')
# For faster runs on limited resources, set steps_per_epoch to a smaller number (e.g. 10)
steps_per_epoch = None

ae.fit(paired_ds, epochs=EPOCHS_AE, callbacks=callbacks)

# Load best
try:
    ae.load_weights(os.path.join(checkpoint_dir, 'ae_best.h5'))
    print('Loaded best AE weights')
except Exception as e:
    print('Could not load weights, continuing with current model. Error:', e)

for batch in ds.take(1):
    clean = batch[:8]
    noisy = add_noise(clean)
    denoised = ae.predict(noisy)
    print('Noisy:')
    show_images(noisy[:8], ncols=8, title='Noisy')
    print('Denoised by AE:')
    show_images(denoised[:8], ncols=8, title='Denoised')
    print('Original:')
    show_images(clean[:8], ncols=8, title='Original')
    break
```

```
Training AE (denoising) ...
Epoch 1/100
104/104 ━━━━━━━━━━ 0s 113ms/step - loss: 0.3459
```

104/104 ━━━━━━━━ 25s 114ms/step - loss: 0.3455 - learning_rate: 1.0000e-04
Epoch 2/100
104/104 ━━━━━━ 0s 77ms/step - loss: 0.1976
104/104 ━━━━ 9s 79ms/step - loss: 0.1974 - learning_rate: 1.0000e-04
Epoch 3/100
103/104 ━━━━ 0s 71ms/step - loss: 0.1277
104/104 ━━━━ 8s 72ms/step - loss: 0.1276 - learning_rate: 1.0000e-04
Epoch 4/100
104/104 ━━━━ 0s 72ms/step - loss: 0.1090
104/104 ━━━━ 11s 74ms/step - loss: 0.1090 - learning_rate: 1.0000e-04
Epoch 5/100
104/104 ━━━━ 0s 85ms/step - loss: 0.0999
104/104 ━━━━ 10s 86ms/step - loss: 0.0999 - learning_rate: 1.0000e-04
Epoch 6/100
104/104 ━━━━ 0s 82ms/step - loss: 0.0941
104/104 ━━━━ 9s 83ms/step - loss: 0.0941 - learning_rate: 1.0000e-04
Epoch 7/100
103/104 ━━━━ 0s 71ms/step - loss: 0.0897
104/104 ━━━━ 8s 72ms/step - loss: 0.0897 - learning_rate: 1.0000e-04
Epoch 8/100
103/104 ━━━━ 0s 78ms/step - loss: 0.0863
104/104 ━━━━ 9s 79ms/step - loss: 0.0863 - learning_rate: 1.0000e-04
Epoch 9/100
104/104 ━━━━ 0s 82ms/step - loss: 0.0845
104/104 ━━━━ 9s 83ms/step - loss: 0.0844 - learning_rate: 1.0000e-04
Epoch 10/100
104/104 ━━━━ 0s 75ms/step - loss: 0.0810
104/104 ━━━━ 9s 77ms/step - loss: 0.0810 - learning_rate: 1.0000e-04
Epoch 11/100
103/104 ━━━━ 0s 77ms/step - loss: 0.0777
104/104 ━━━━ 9s 77ms/step - loss: 0.0777 - learning_rate: 1.0000e-04
Epoch 12/100
103/104 ━━━━ 0s 82ms/step - loss: 0.0759
104/104 ━━━━ 9s 83ms/step - loss: 0.0759 - learning_rate: 1.0000e-04
Epoch 13/100
103/104 ━━━━ 0s 82ms/step - loss: 0.0728
104/104 ━━━━ 9s 82ms/step - loss: 0.0728 - learning_rate: 1.0000e-04
Epoch 14/100
103/104 ━━━━ 0s 71ms/step - loss: 0.0704
104/104 ━━━━ 8s 72ms/step - loss: 0.0704 - learning_rate: 1.0000e-04
Epoch 15/100
103/104 ━━━━ 0s 71ms/step - loss: 0.0684
104/104 ━━━━ 9s 71ms/step - loss: 0.0684 - learning_rate: 1.0000e-04
Epoch 16/100
103/104 ━━━━ 0s 82ms/step - loss: 0.0664
104/104 ━━━━ 11s 82ms/step - loss: 0.0664 - learning_rate: 1.0000e-04
Epoch 17/100
103/104 ━━━━ 0s 81ms/step - loss: 0.0647
104/104 ━━━━ 9s 81ms/step - loss: 0.0647 - learning_rate: 1.0000e-04
Epoch 18/100
104/104 ━━━━ 0s 70ms/step - loss: 0.0636

104/104 ————— 8s 72ms/step - loss: 0.0636 - learning_rate: 1.0000e-04
Epoch 19/100
103/104 ————— 0s 73ms/step - loss: 0.0625
104/104 ————— 9s 73ms/step - loss: 0.0625 - learning_rate: 1.0000e-04
Epoch 20/100
104/104 ————— 0s 82ms/step - loss: 0.0612
104/104 ————— 9s 83ms/step - loss: 0.0612 - learning_rate: 1.0000e-04
Epoch 21/100
103/104 ————— 0s 81ms/step - loss: 0.0601
104/104 ————— 9s 82ms/step - loss: 0.0601 - learning_rate: 1.0000e-04
Epoch 22/100
103/104 ————— 0s 72ms/step - loss: 0.0599
104/104 ————— 8s 72ms/step - loss: 0.0599 - learning_rate: 1.0000e-04
Epoch 23/100
103/104 ————— 0s 81ms/step - loss: 0.0586
104/104 ————— 9s 82ms/step - loss: 0.0586 - learning_rate: 1.0000e-04
Epoch 24/100
103/104 ————— 0s 82ms/step - loss: 0.0582
104/104 ————— 9s 83ms/step - loss: 0.0582 - learning_rate: 1.0000e-04
Epoch 25/100
103/104 ————— 0s 79ms/step - loss: 0.0574
104/104 ————— 9s 80ms/step - loss: 0.0574 - learning_rate: 1.0000e-04
Epoch 26/100
104/104 ————— 0s 72ms/step - loss: 0.0568
104/104 ————— 9s 73ms/step - loss: 0.0568 - learning_rate: 1.0000e-04
Epoch 27/100
103/104 ————— 0s 83ms/step - loss: 0.0563
104/104 ————— 10s 83ms/step - loss: 0.0563 - learning_rate: 1.0000e-04
4
Epoch 28/100
103/104 ————— 0s 83ms/step - loss: 0.0554
104/104 ————— 9s 83ms/step - loss: 0.0554 - learning_rate: 1.0000e-04
Epoch 29/100
103/104 ————— 0s 73ms/step - loss: 0.0548
104/104 ————— 9s 74ms/step - loss: 0.0548 - learning_rate: 1.0000e-04
Epoch 30/100
104/104 ————— 0s 72ms/step - loss: 0.0540
104/104 ————— 9s 73ms/step - loss: 0.0540 - learning_rate: 1.0000e-04
Epoch 31/100
103/104 ————— 0s 81ms/step - loss: 0.0537
104/104 ————— 10s 82ms/step - loss: 0.0537 - learning_rate: 1.0000e-04
4
Epoch 32/100
103/104 ————— 0s 82ms/step - loss: 0.0534
104/104 ————— 9s 83ms/step - loss: 0.0534 - learning_rate: 1.0000e-04
Epoch 33/100
103/104 ————— 0s 74ms/step - loss: 0.0529
104/104 ————— 9s 75ms/step - loss: 0.0529 - learning_rate: 1.0000e-04
Epoch 34/100
104/104 ————— 0s 72ms/step - loss: 0.0528
104/104 ————— 10s 74ms/step - loss: 0.0528 - learning_rate: 1.0000e-04
4
Epoch 35/100
103/104 ————— 0s 82ms/step - loss: 0.0521

104/104 ━━━━━━━━ 10s 83ms/step - loss: 0.0521 - learning_rate: 1.0000e-04

Epoch 36/100

104/104 ━━━━━━ 0s 81ms/step - loss: 0.0518

104/104 ━━━━ 9s 82ms/step - loss: 0.0518 - learning_rate: 1.0000e-04

Epoch 37/100

103/104 ━━━━ 0s 82ms/step - loss: 0.0516

104/104 ━━━━ 9s 83ms/step - loss: 0.0516 - learning_rate: 1.0000e-04

Epoch 38/100

103/104 ━━━━ 0s 73ms/step - loss: 0.0508

104/104 ━━━━ 9s 74ms/step - loss: 0.0508 - learning_rate: 1.0000e-04

Epoch 39/100

104/104 ━━━━ 0s 85ms/step - loss: 0.0507

104/104 ━━━━ 10s 86ms/step - loss: 0.0507 - learning_rate: 1.0000e-04

4

Epoch 40/100

103/104 ━━━━ 0s 82ms/step - loss: 0.0502

104/104 ━━━━ 9s 83ms/step - loss: 0.0502 - learning_rate: 1.0000e-04

Epoch 41/100

103/104 ━━━━ 0s 77ms/step - loss: 0.0500

104/104 ━━━━ 9s 78ms/step - loss: 0.0500 - learning_rate: 1.0000e-04

Epoch 42/100

103/104 ━━━━ 0s 74ms/step - loss: 0.0496

104/104 ━━━━ 9s 74ms/step - loss: 0.0496 - learning_rate: 1.0000e-04

Epoch 43/100

104/104 ━━━━ 0s 83ms/step - loss: 0.0492

104/104 ━━━━ 10s 84ms/step - loss: 0.0492 - learning_rate: 1.0000e-04

4

Epoch 44/100

104/104 ━━━━ 0s 82ms/step - loss: 0.0489

104/104 ━━━━ 9s 83ms/step - loss: 0.0489 - learning_rate: 1.0000e-04

Epoch 45/100

103/104 ━━━━ 0s 72ms/step - loss: 0.0486

104/104 ━━━━ 8s 73ms/step - loss: 0.0486 - learning_rate: 1.0000e-04

Epoch 46/100

103/104 ━━━━ 0s 72ms/step - loss: 0.0487

104/104 ━━━━ 9s 73ms/step - loss: 0.0487 - learning_rate: 1.0000e-04

Epoch 47/100

103/104 ━━━━ 0s 81ms/step - loss: 0.0482

104/104 ━━━━ 9s 81ms/step - loss: 0.0482 - learning_rate: 1.0000e-04

Epoch 48/100

104/104 ━━━━ 0s 81ms/step - loss: 0.0477

104/104 ━━━━ 9s 82ms/step - loss: 0.0477 - learning_rate: 1.0000e-04

Epoch 49/100

104/104 ━━━━ 0s 71ms/step - loss: 0.0475

104/104 ━━━━ 8s 72ms/step - loss: 0.0475 - learning_rate: 1.0000e-04

Epoch 50/100

103/104 ━━━━ 0s 72ms/step - loss: 0.0473

103/104 ━━━━━━━━ 0s 72ms/step - loss: 0.0473

104/104 ━━━━ 10s 73ms/step - loss: 0.0473 - learning_rate: 1.0000e-04

4

Epoch 51/100

103/104 ━━━━ 0s 83ms/step - loss: 0.0468

104/104 ━━━━━━ 9s 83ms/step - loss: 0.0468 - learning_rate: 1.0000e-04
Epoch 52/100
104/104 ━━━━━━ 0s 83ms/step - loss: 0.0473
104/104 ━━━━━━ 10s 85ms/step - loss: 0.0473 - learning_rate: 1.0000e-04
4
Epoch 53/100
103/104 ━━━━ 0s 73ms/step - loss: 0.0466
104/104 ━━━━━━ 8s 74ms/step - loss: 0.0466 - learning_rate: 1.0000e-04
Epoch 54/100
103/104 ━━━━ 0s 73ms/step - loss: 0.0463
104/104 ━━━━━━ 9s 73ms/step - loss: 0.0463 - learning_rate: 1.0000e-04
Epoch 55/100
104/104 ━━━━━━ 0s 82ms/step - loss: 0.0463
104/104 ━━━━━━ 9s 83ms/step - loss: 0.0463 - learning_rate: 1.0000e-04
Epoch 56/100
103/104 ━━━━ 0s 82ms/step - loss: 0.0457
104/104 ━━━━━━ 9s 83ms/step - loss: 0.0458 - learning_rate: 1.0000e-04
Epoch 57/100
103/104 ━━━━ 0s 73ms/step - loss: 0.0457
104/104 ━━━━━━ 8s 73ms/step - loss: 0.0457 - learning_rate: 1.0000e-04
Epoch 58/100
103/104 ━━━━ 0s 73ms/step - loss: 0.0455
104/104 ━━━━━━ 10s 74ms/step - loss: 0.0455 - learning_rate: 1.0000e-04
4
Epoch 59/100
104/104 ━━━━━━ 0s 72ms/step - loss: 0.0454
104/104 ━━━━━━ 11s 73ms/step - loss: 0.0454 - learning_rate: 1.0000e-04
4
Epoch 60/100
103/104 ━━━━ 0s 82ms/step - loss: 0.0453
104/104 ━━━━━━ 11s 83ms/step - loss: 0.0453 - learning_rate: 1.0000e-04
4
Epoch 61/100
104/104 ━━━━━━ 0s 83ms/step - loss: 0.0452
104/104 ━━━━━━ 10s 84ms/step - loss: 0.0452 - learning_rate: 1.0000e-04
4
Epoch 62/100
103/104 ━━━━ 0s 78ms/step - loss: 0.0448
104/104 ━━━━━━ 9s 79ms/step - loss: 0.0448 - learning_rate: 1.0000e-04
Epoch 63/100
103/104 ━━━━ 0s 72ms/step - loss: 0.0445

104/104 ————— 9s 75ms/step - loss: 0.0430 - learning_rate: 1.0000e-04
Epoch 76/100
103/104 ————— 0s 80ms/step - loss: 0.0427
104/104 ————— 10s 81ms/step - loss: 0.0427 - learning_rate: 1.0000e-04
Epoch 77/100
104/104 ————— 0s 84ms/step - loss: 0.0425
104/104 ————— 10s 85ms/step - loss: 0.0425 - learning_rate: 1.0000e-04
Epoch 78/100
104/104 ————— 0s 83ms/step - loss: 0.0425
104/104 ————— 10s 84ms/step - loss: 0.0425 - learning_rate: 1.0000e-04
Epoch 79/100
103/104 ————— 0s 73ms/step - loss: 0.0423
104/104 ————— 9s 74ms/step - loss: 0.0423 - learning_rate: 1.0000e-04
Epoch 80/100
103/104 ————— 0s 73ms/step - loss: 0.0424
104/104 ————— 9s 74ms/step - loss: 0.0424 - learning_rate: 1.0000e-04
Epoch 81/100
104/104 ————— 0s 83ms/step - loss: 0.0420
104/104 ————— 10s 84ms/step - loss: 0.0420 - learning_rate: 1.0000e-04
Epoch 82/100
104/104 ————— 0s 83ms/step - loss: 0.0422
104/104 ————— 10s 84ms/step - loss: 0.0422 - learning_rate: 1.0000e-04
Epoch 83/100
104/104 ————— 8s 74ms/step - loss: 0.0420 - learning_rate: 1.0000e-04
Epoch 84/100
103/104 ————— 0s 76ms/step - loss: 0.0418
104/104 ————— 10s 77ms/step - loss: 0.0418 - learning_rate: 1.0000e-04
Epoch 85/100
103/104 ————— 0s 83ms/step - loss: 0.0419
104/104 ————— 9s 83ms/step - loss: 0.0419 - learning_rate: 1.0000e-04
Epoch 86/100
103/104 ————— 0s 84ms/step - loss: 0.0415
104/104 ————— 10s 84ms/step - loss: 0.0415 - learning_rate: 1.0000e-04
Epoch 87/100
103/104 ————— 0s 73ms/step - loss: 0.0414

104/104 9s 74ms/step - loss: 0.0414 - learning_rate: 1.0000e-04
Epoch 88/100
104/104 0s 73ms/step - loss: 0.0414
104/104 10s 74ms/step - loss: 0.0414 - learning_rate: 1.0000e-04
4
Epoch 89/100
103/104 0s 83ms/step - loss: 0.0417
104/104 9s 84ms/step - loss: 0.0417 - learning_rate: 1.0000e-04
Epoch 90/100
103/104 0s 83ms/step - loss: 0.0413
104/104 10s 84ms/step - loss: 0.0413 - learning_rate: 1.0000e-04
4
Epoch 91/100
103/104 0s 76ms/step - loss: 0.0413
104/104 9s 77ms/step - loss: 0.0413 - learning_rate: 1.0000e-04
Epoch 92/100
104/104 0s 73ms/step - loss: 0.0411
104/104 9s 75ms/step - loss: 0.0411 - learning_rate: 1.0000e-04
Epoch 93/100
103/104 0s 81ms/step - loss: 0.0413
104/104 11s 82ms/step - loss: 0.0413 - learning_rate: 1.0000e-04
4
Epoch 94/100
104/104 0s 82ms/step - loss: 0.0411
104/104 9s 84ms/step - loss: 0.0411 - learning_rate: 1.0000e-04
Epoch 95/100
104/104 0s 82ms/step - loss: 0.0408
104/104 10s 83ms/step - loss: 0.0408 - learning_rate: 1.0000e-04
4
Epoch 96/100
104/104 0s 74ms/step - loss: 0.0409
104/104 9s 75ms/step - loss: 0.0409 - learning_rate: 1.0000e-04
Epoch 97/100
104/104 0s 83ms/step - loss: 0.0406
Epoch 97: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.
104/104 10s 85ms/step - loss: 0.0406 - learning_rate: 1.0000e-04
4
Epoch 98/100
104/104 0s 85ms/step - loss: 0.0405
104/104 10s 86ms/step - loss: 0.0405 - learning_rate: 5.0000e-05
5
Epoch 99/100
104/104 0s 84ms/step - loss: 0.0402

```

104/104 ━━━━━━━━━━━━━━━━ 10s 85ms/step - loss: 0.0402 - learning_rate: 5.0000e-0
5
Epoch 100/100
104/104 ━━━━━━━━━━━━━━━━ 0s 72ms/step - loss: 0.0404
104/104 ━━━━━━━━━━━━━━━━ 8s 73ms/step - loss: 0.0404 - learning_rate: 5.0000e-05
Loaded best AE weights
1/1 ━━━━━━━ 1s 1s/step
Noisy:

```



Denoised by AE:



Original:



Build a Convolutional Variational Autoencoder (VAE) for face generation & Train the VAE

```

In [ ]: class Sampling(layers.Layer):
    def call(self, inputs):
        z_mean, z_log_var = inputs
        batch = tf.shape(z_mean)[0]
        dim = tf.shape(z_mean)[1]
        epsilon = tf.random.normal(shape=(batch, dim))
        return z_mean + tf.exp(0.5 * z_log_var) * epsilon

# Encoder for VAE
vae_encoder_inputs = layers.Input(shape=(IMAGE_SIZE, IMAGE_SIZE, 3))

x = layers.Conv2D(32, 3, strides=2, padding='same', activation='relu')(vae_encoder_
x = layers.Conv2D(64, 3, strides=2, padding='same', activation='relu')(x) # 16x16
x = layers.Conv2D(128, 3, strides=2, padding='same', activation='relu')(x) # 8x8
x = layers.Conv2D(256, 3, strides=2, padding='same', activation='relu')(x) # 4x4
x = layers.Flatten()(x)
vae_x = layers.Dense(512, activation='relu')(x)
z_mean = layers.Dense(LATENT_DIM, name='z_mean')(vae_x)
z_log_var = layers.Dense(LATENT_DIM, name='z_log_var')(vae_x)
z = Sampling()([z_mean, z_log_var])
vae_encoder = keras.Model(vae_encoder_inputs, [z_mean, z_log_var, z], name='vae_enc
vae_encoder.summary()

```

```

# Decoder for VAE (re-using architecture pattern from AE)
latent_inputs = layers.Input(shape=(LATENT_DIM,))
x = layers.Dense(np.prod(shape_before_flatten), activation='relu')(latent_inputs)
x = layers.Reshape(shape_before_flatten)(x)
x = layers.Conv2DTranspose(256, 3, strides=2, padding='same', activation='relu')(x)
x = layers.Conv2DTranspose(128, 3, strides=2, padding='same', activation='relu')(x)
x = layers.Conv2DTranspose(64, 3, strides=2, padding='same', activation='relu')(x)
x = layers.Conv2DTranspose(32, 3, strides=2, padding='same', activation='relu')(x)
vae_outputs = layers.Conv2D(3, 3, activation='tanh', padding='same')(x)
vae_decoder = keras.Model(latent_inputs, vae_outputs, name='vae_decoder')
vae_decoder.summary()

# VAE model with custom train_step
class VAE(keras.Model):
    def __init__(self, encoder, decoder, **kwargs):
        super(VAE, self).__init__(**kwargs)
        self.encoder = encoder
        self.decoder = decoder
        self.total_loss_tracker = keras.metrics.Mean(name="total_loss")
        self.reconstruction_loss_tracker = keras.metrics.Mean(name="reconstruction_loss")
        self.kl_loss_tracker = keras.metrics.Mean(name="kl_loss")

    @property
    def metrics(self):
        return [self.total_loss_tracker, self.reconstruction_loss_tracker, self.kl_loss_tracker]

    def train_step(self, data):
        if isinstance(data, tuple):
            data = data[0]
        with tf.GradientTape() as tape:
            z_mean, z_log_var, z = self.encoder(data)
            reconstruction = self.decoder(z)
            reconstruction_loss = tf.reduce_mean(tf.reduce_sum(tf.square(data - reconstruction), axis=[1, 2, 3]))
            kl_loss = -0.5 * tf.reduce_mean(tf.reduce_sum(1 + z_log_var - tf.square(z_mean) - tf.exp(z_log_var), axis=1))
            total_loss = reconstruction_loss + kl_loss * 1.0
        grads = tape.gradient(total_loss, self.trainable_weights)
        self.optimizer.apply_gradients(zip(grads, self.trainable_weights))
        self.total_loss_tracker.update_state(total_loss)
        self.reconstruction_loss_tracker.update_state(reconstruction_loss)
        self.kl_loss_tracker.update_state(kl_loss)
        return {
            "loss": self.total_loss_tracker.result(),
            "reconstruction_loss": self.reconstruction_loss_tracker.result(),
            "kl_loss": self.kl_loss_tracker.result(),
        }

vae = VAE(vae_encoder, vae_decoder)
vae.compile(optimizer=keras.optimizers.Adam(1e-4))

checkpoint_dir_vae = '/content/checkpoints_vae'
os.makedirs(checkpoint_dir_vae, exist_ok=True)

callbacks_vae = [
    keras.callbacks.ModelCheckpoint(os.path.join(checkpoint_dir_vae, 'vae_best.h5'))
]

```

```

        keras.callbacks.ReduceLROnPlateau(monitor='loss', factor=0.5, patience=4, verbose=0)

    print('Training VAE ...')
    vae.fit(ds, epochs=EPOCHS_VAE, callbacks=callbacks_vae)

try:
    vae.load_weights(os.path.join(checkpoint_dir_vae, 'vae_best.h5'))
    print('Loaded best VAE weights')
except Exception as e:
    print('Could not load VAE weights, continuing. Error:', e)

```

Model: "vae_encoder"

Layer (type)	Output Shape	Param #	Connected to
input_layer_3 (InputLayer)	(None, 64, 64, 3)	0	-
conv2d_7 (Conv2D)	(None, 32, 32, 32)	896	input_layer_3[0]...
conv2d_8 (Conv2D)	(None, 16, 16, 64)	18,496	conv2d_7[0][0]
conv2d_9 (Conv2D)	(None, 8, 8, 128)	73,856	conv2d_8[0][0]
conv2d_10 (Conv2D)	(None, 4, 4, 256)	295,168	conv2d_9[0][0]
flatten_2 (Flatten)	(None, 4096)	0	conv2d_10[0][0]
dense_2 (Dense)	(None, 512)	2,097,664	flatten_2[0][0]
z_mean (Dense)	(None, 128)	65,664	dense_2[0][0]
z_log_var (Dense)	(None, 128)	65,664	dense_2[0][0]
sampling (Sampling)	(None, 128)	0	z_mean[0][0], z_log_var[0][0]

Total params: 2,617,408 (9.98 MB)

Trainable params: 2,617,408 (9.98 MB)

Non-trainable params: 0 (0.00 B)

Model: "vae_decoder"

Layer (type)	Output Shape	Param #
input_layer_4 (InputLayer)	(None, 128)	0
dense_3 (Dense)	(None, 4096)	528,384
reshape_2 (Reshape)	(None, 4, 4, 256)	0
conv2d_transpose_7 (Conv2DTranspose)	(None, 8, 8, 256)	590,080
conv2d_transpose_8 (Conv2DTranspose)	(None, 16, 16, 128)	295,040
conv2d_transpose_9 (Conv2DTranspose)	(None, 32, 32, 64)	73,792
conv2d_transpose_10 (Conv2DTranspose)	(None, 64, 64, 32)	18,464
conv2d_11 (Conv2D)	(None, 64, 64, 3)	867

Total params: 1,506,627 (5.75 MB)

Trainable params: 1,506,627 (5.75 MB)

Non-trainable params: 0 (0.00 B)

Training VAE ...

Epoch 1/100

104/104 ————— 0s 98ms/step - kl_loss: 27.9567 - loss: 4033.0061 - reconstruction_loss: 4005.0493

104/104 ————— 17s 99ms/step - kl_loss: 28.1562 - loss: 4028.1794 - reconstruction_loss: 4000.0232 - learning_rate: 1.0000e-04

Epoch 2/100

103/104 ————— 0s 69ms/step - kl_loss: 66.3656 - loss: 2593.8445 - reconstruction_loss: 2527.4788

104/104 ————— 8s 70ms/step - kl_loss: 66.3978 - loss: 2591.4243 - reconstruction_loss: 2525.0261 - learning_rate: 1.0000e-04

Epoch 3/100

103/104 ————— 0s 58ms/step - kl_loss: 98.7794 - loss: 1986.3716 - reconstruction_loss: 1887.5920

104/104 ————— 7s 60ms/step - kl_loss: 99.0076 - loss: 1983.5228 - reconstruction_loss: 1884.5151 - learning_rate: 1.0000e-04

Epoch 4/100

103/104 ————— 0s 58ms/step - kl_loss: 124.2774 - loss: 1588.5659 - reconstruction_loss: 1464.2885

104/104 ————— 8s 59ms/step - kl_loss: 124.3116 - loss: 1587.8683 - reconstruction_loss: 1463.5565 - learning_rate: 1.0000e-04

Epoch 5/100

103/104 ————— 0s 66ms/step - kl_loss: 130.0995 - loss: 1459.0902 - reconstruction_loss: 1328.9906

104/104 ————— 8s 67ms/step - kl_loss: 130.1177 - loss: 1458.8474 - re
construction_loss: 1328.7296 - learning_rate: 1.0000e-04
Epoch 6/100
104/104 ————— 0s 57ms/step - kl_loss: 134.6544 - loss: 1379.2657 - re
construction_loss: 1244.6112
104/104 ————— 9s 59ms/step - kl_loss: 134.6695 - loss: 1379.1772 - re
construction_loss: 1244.5077 - learning_rate: 1.0000e-04
Epoch 7/100
103/104 ————— 0s 58ms/step - kl_loss: 141.3997 - loss: 1315.2732 - re
construction_loss: 1173.8735
104/104 ————— 8s 59ms/step - kl_loss: 141.4241 - loss: 1315.0986 - re
construction_loss: 1173.6746 - learning_rate: 1.0000e-04
Epoch 8/100
103/104 ————— 0s 67ms/step - kl_loss: 145.1208 - loss: 1260.4790 - re
construction_loss: 1115.3582
104/104 ————— 8s 68ms/step - kl_loss: 145.1466 - loss: 1260.3247 - re
construction_loss: 1115.1781 - learning_rate: 1.0000e-04
Epoch 9/100
104/104 ————— 0s 57ms/step - kl_loss: 147.3746 - loss: 1218.8527 - re
construction_loss: 1071.4783
104/104 ————— 9s 59ms/step - kl_loss: 147.3836 - loss: 1218.7856 - re
construction_loss: 1071.4022 - learning_rate: 1.0000e-04
Epoch 10/100
103/104 ————— 0s 57ms/step - kl_loss: 150.8293 - loss: 1175.6901 - re
construction_loss: 1024.8607
104/104 ————— 8s 58ms/step - kl_loss: 150.8398 - loss: 1175.5842 - re
construction_loss: 1024.7444 - learning_rate: 1.0000e-04
Epoch 11/100
103/104 ————— 0s 66ms/step - kl_loss: 152.7605 - loss: 1136.8182 - re
construction_loss: 984.0575
104/104 ————— 8s 67ms/step - kl_loss: 152.7806 - loss: 1136.7644 - re
construction_loss: 983.9835 - learning_rate: 1.0000e-04
Epoch 12/100
103/104 ————— 0s 56ms/step - kl_loss: 154.4276 - loss: 1107.1888 - re
construction_loss: 952.7614
104/104 ————— 7s 57ms/step - kl_loss: 154.4474 - loss: 1107.1342 - re
construction_loss: 952.6869 - learning_rate: 1.0000e-04
Epoch 13/100
103/104 ————— 0s 66ms/step - kl_loss: 157.0170 - loss: 1084.0210 - re
construction_loss: 927.0041
104/104 ————— 8s 67ms/step - kl_loss: 157.0199 - loss: 1083.9308 - re
construction_loss: 926.9110 - learning_rate: 1.0000e-04
Epoch 14/100
104/104 ————— 0s 56ms/step - kl_loss: 158.6265 - loss: 1063.1409 - re
construction_loss: 904.5143
104/104 ————— 7s 58ms/step - kl_loss: 158.6293 - loss: 1063.1110 - re
construction_loss: 904.4816 - learning_rate: 1.0000e-04
Epoch 15/100
103/104 ————— 0s 58ms/step - kl_loss: 158.5908 - loss: 1048.5592 - re
construction_loss: 889.9685
104/104 ————— 10s 59ms/step - kl_loss: 158.6093 - loss: 1048.4855 - r
econstruction_loss: 889.8763 - learning_rate: 1.0000e-04
Epoch 16/100
103/104 ————— 0s 69ms/step - kl_loss: 159.3213 - loss: 1029.2008 - re
construction_loss: 869.8796

104/104 ————— 8s 70ms/step - kl_loss: 159.3344 - loss: 1029.1396 - reconstruction_loss: 869.8054 - learning_rate: 1.0000e-04
Epoch 17/100
103/104 ————— 0s 59ms/step - kl_loss: 160.6732 - loss: 1020.7712 - reconstruction_loss: 860.0981
104/104 ————— 7s 61ms/step - kl_loss: 160.6769 - loss: 1020.6748 - reconstruction_loss: 859.9979 - learning_rate: 1.0000e-04
Epoch 18/100
103/104 ————— 0s 59ms/step - kl_loss: 160.9794 - loss: 1005.2012 - reconstruction_loss: 844.2218
104/104 ————— 7s 60ms/step - kl_loss: 160.9842 - loss: 1005.1525 - reconstruction_loss: 844.1683 - learning_rate: 1.0000e-04
Epoch 19/100
103/104 ————— 0s 68ms/step - kl_loss: 161.7342 - loss: 993.7817 - reconstruction_loss: 832.0475
104/104 ————— 8s 69ms/step - kl_loss: 161.7338 - loss: 993.7390 - reconstruction_loss: 832.0051 - learning_rate: 1.0000e-04
Epoch 20/100
103/104 ————— 0s 59ms/step - kl_loss: 162.1061 - loss: 981.6159 - reconstruction_loss: 819.5099
104/104 ————— 7s 60ms/step - kl_loss: 162.1134 - loss: 981.5817 - reconstruction_loss: 819.4684 - learning_rate: 1.0000e-04
Epoch 21/100
103/104 ————— 0s 68ms/step - kl_loss: 162.7362 - loss: 970.2355 - reconstruction_loss: 807.4992
104/104 ————— 8s 69ms/step - kl_loss: 162.7368 - loss: 970.2155 - reconstruction_loss: 807.4785 - learning_rate: 1.0000e-04
Epoch 22/100
103/104 ————— 0s 67ms/step - kl_loss: 162.4373 - loss: 960.5093 - reconstruction_loss: 798.0719
104/104 ————— 10s 68ms/step - kl_loss: 162.4510 - loss: 960.5018 - reconstruction_loss: 798.0507 - learning_rate: 1.0000e-04
Epoch 23/100
103/104 ————— 0s 57ms/step - kl_loss: 162.5886 - loss: 959.9982 - reconstruction_loss: 797.4098
104/104 ————— 7s 59ms/step - kl_loss: 162.5995 - loss: 959.9368 - reconstruction_loss: 797.3375 - learning_rate: 1.0000e-04
Epoch 24/100
103/104 ————— 0s 68ms/step - kl_loss: 163.2363 - loss: 943.4771 - reconstruction_loss: 780.2407
104/104 ————— 8s 69ms/step - kl_loss: 163.2478 - loss: 943.5070 - reconstruction_loss: 780.2590 - learning_rate: 1.0000e-04
Epoch 25/100
104/104 ————— 0s 59ms/step - kl_loss: 164.2300 - loss: 936.5526 - reconstruction_loss: 772.3223
104/104 ————— 7s 60ms/step - kl_loss: 164.2306 - loss: 936.5493 - reconstruction_loss: 772.3185 - learning_rate: 1.0000e-04
Epoch 26/100
103/104 ————— 0s 58ms/step - kl_loss: 164.0498 - loss: 930.8279 - reconstruction_loss: 766.7781
104/104 ————— 7s 59ms/step - kl_loss: 164.0575 - loss: 930.8151 - reconstruction_loss: 766.7576 - learning_rate: 1.0000e-04
Epoch 27/100
103/104 ————— 0s 67ms/step - kl_loss: 164.3585 - loss: 923.5911 - reconstruction_loss: 759.2324

104/104 ————— 8s 68ms/step - kl_loss: 164.3653 - loss: 923.5706 - reconstruction_loss: 759.2052 - learning_rate: 1.0000e-04
Epoch 28/100
103/104 ————— 0s 58ms/step - kl_loss: 164.1848 - loss: 917.2325 - reconstruction_loss: 753.0476
104/104 ————— 7s 59ms/step - kl_loss: 164.1932 - loss: 917.2386 - reconstruction_loss: 753.0453 - learning_rate: 1.0000e-04
Epoch 29/100
103/104 ————— 0s 58ms/step - kl_loss: 165.1221 - loss: 911.9058 - reconstruction_loss: 746.7837
104/104 ————— 10s 59ms/step - kl_loss: 165.1259 - loss: 911.9073 - reconstruction_loss: 746.7814 - learning_rate: 1.0000e-04
Epoch 30/100
103/104 ————— 0s 68ms/step - kl_loss: 164.9326 - loss: 906.5822 - reconstruction_loss: 741.6497
104/104 ————— 8s 69ms/step - kl_loss: 164.9379 - loss: 906.5524 - reconstruction_loss: 741.6146 - learning_rate: 1.0000e-04
Epoch 31/100
103/104 ————— 0s 59ms/step - kl_loss: 165.5345 - loss: 901.6142 - reconstruction_loss: 736.0796
104/104 ————— 7s 60ms/step - kl_loss: 165.5363 - loss: 901.5569 - reconstruction_loss: 736.0206 - learning_rate: 1.0000e-04
Epoch 32/100
103/104 ————— 0s 62ms/step - kl_loss: 165.3780 - loss: 894.8919 - reconstruction_loss: 729.5139
104/104 ————— 8s 63ms/step - kl_loss: 165.3821 - loss: 894.8796 - reconstruction_loss: 729.4974 - learning_rate: 1.0000e-04
Epoch 33/100
104/104 ————— 0s 68ms/step - kl_loss: 165.5012 - loss: 890.6415 - reconstruction_loss: 725.1403
104/104 ————— 8s 69ms/step - kl_loss: 165.5021 - loss: 890.6318 - reconstruction_loss: 725.1297 - learning_rate: 1.0000e-04
Epoch 34/100
104/104 ————— 0s 58ms/step - kl_loss: 165.5283 - loss: 883.7618 - reconstruction_loss: 718.2336
104/104 ————— 7s 59ms/step - kl_loss: 165.5318 - loss: 883.7773 - reconstruction_loss: 718.2457 - learning_rate: 1.0000e-04
Epoch 35/100
103/104 ————— 0s 68ms/step - kl_loss: 165.9801 - loss: 878.5512 - reconstruction_loss: 712.5710
104/104 ————— 8s 69ms/step - kl_loss: 165.9817 - loss: 878.5447 - reconstruction_loss: 712.5629 - learning_rate: 1.0000e-04
Epoch 36/100
103/104 ————— 0s 59ms/step - kl_loss: 166.1290 - loss: 874.8445 - reconstruction_loss: 708.7156
104/104 ————— 7s 60ms/step - kl_loss: 166.1322 - loss: 874.8552 - reconstruction_loss: 708.7231 - learning_rate: 1.0000e-04
Epoch 37/100
103/104 ————— 0s 60ms/step - kl_loss: 165.8461 - loss: 870.4674 - reconstruction_loss: 704.6213
104/104 ————— 8s 61ms/step - kl_loss: 165.8507 - loss: 870.4664 - reconstruction_loss: 704.6157 - learning_rate: 1.0000e-04
Epoch 38/100
103/104 ————— 0s 68ms/step - kl_loss: 166.2821 - loss: 866.0134 - reconstruction_loss: 699.7313

104/104 ————— 8s 69ms/step - kl_loss: 166.2871 - loss: 866.0010 - reconstruction_loss: 699.7139 - learning_rate: 1.0000e-04
Epoch 39/100
103/104 ————— 0s 58ms/step - kl_loss: 165.9303 - loss: 861.9463 - reconstruction_loss: 696.0160
104/104 ————— 7s 59ms/step - kl_loss: 165.9371 - loss: 861.9458 - reconstruction_loss: 696.0086 - learning_rate: 1.0000e-04
Epoch 40/100
103/104 ————— 0s 69ms/step - kl_loss: 166.3261 - loss: 855.9739 - reconstruction_loss: 689.6478
104/104 ————— 8s 70ms/step - kl_loss: 166.3273 - loss: 856.0031 - reconstruction_loss: 689.6757 - learning_rate: 1.0000e-04
Epoch 41/100
104/104 ————— 7s 58ms/step - kl_loss: 166.1965 - loss: 854.5685 - reconstruction_loss: 688.3719 - learning_rate: 1.0000e-04
Epoch 42/100
103/104 ————— 0s 60ms/step - kl_loss: 166.9865 - loss: 851.3837 - reconstruction_loss: 684.3972
104/104 ————— 8s 61ms/step - kl_loss: 166.9864 - loss: 851.3532 - reconstruction_loss: 684.3668 - learning_rate: 1.0000e-04
Epoch 43/100
103/104 ————— 0s 69ms/step - kl_loss: 166.3796 - loss: 847.6194 - reconstruction_loss: 681.2399
104/104 ————— 8s 69ms/step - kl_loss: 166.3885 - loss: 847.6169 - reconstruction_loss: 681.2285 - learning_rate: 1.0000e-04
Epoch 44/100
103/104 ————— 0s 58ms/step - kl_loss: 166.1850 - loss: 838.6499 - reconstruction_loss: 672.4650
104/104 ————— 7s 59ms/step - kl_loss: 166.2008 - loss: 838.7276 - reconstruction_loss: 672.5269 - learning_rate: 1.0000e-04
Epoch 45/100
103/104 ————— 0s 68ms/step - kl_loss: 166.6644 - loss: 842.6873 - reconstruction_loss: 676.0228
104/104 ————— 8s 69ms/step - kl_loss: 166.6702 - loss: 842.6483 - reconstruction_loss: 675.9779 - learning_rate: 1.0000e-04
Epoch 46/100
103/104 ————— 0s 58ms/step - kl_loss: 166.7663 - loss: 836.9667 - reconstruction_loss: 670.2004
104/104 ————— 7s 59ms/step - kl_loss: 166.7697 - loss: 836.9867 - reconstruction_loss: 670.2170 - learning_rate: 1.0000e-04
Epoch 47/100
103/104 ————— 0s 59ms/step - kl_loss: 167.4197 - loss: 834.5900 - reconstruction_loss: 667.1701
104/104 ————— 8s 60ms/step - kl_loss: 167.4153 - loss: 834.5538 - reconstruction_loss: 667.1383 - learning_rate: 1.0000e-04
Epoch 48/100
103/104 ————— 0s 69ms/step - kl_loss: 167.8369 - loss: 833.8884 - reconstruction_loss: 666.0515
104/104 ————— 8s 70ms/step - kl_loss: 167.8291 - loss: 833.8551 - reconstruction_loss: 666.0259 - learning_rate: 1.0000e-04
Epoch 49/100
104/104 ————— 0s 59ms/step - kl_loss: 167.1550 - loss: 828.5118 - reconstruction_loss: 661.3568

104/104 ————— 9s 60ms/step - kl_loss: 167.1588 - loss: 828.5067 - reconstruction_loss: 661.3479 - learning_rate: 1.0000e-04
Epoch 50/100
103/104 ————— 0s 58ms/step - kl_loss: 167.3493 - loss: 823.1793 - reconstruction_loss: 655.8301
104/104 ————— 10s 60ms/step - kl_loss: 167.3545 - loss: 823.2445 - reconstruction_loss: 655.8900 - learning_rate: 1.0000e-04
Epoch 51/100
103/104 ————— 0s 68ms/step - kl_loss: 167.3209 - loss: 821.6088 - reconstruction_loss: 654.2879
104/104 ————— 8s 69ms/step - kl_loss: 167.3247 - loss: 821.5888 - reconstruction_loss: 654.2641 - learning_rate: 1.0000e-04
Epoch 52/100
103/104 ————— 0s 61ms/step - kl_loss: 167.7442 - loss: 818.5480 - reconstruction_loss: 650.8038
104/104 ————— 7s 63ms/step - kl_loss: 167.7469 - loss: 818.5707 - reconstruction_loss: 650.8238 - learning_rate: 1.0000e-04
Epoch 53/100
103/104 ————— 0s 58ms/step - kl_loss: 167.5050 - loss: 821.8288 - reconstruction_loss: 654.3240
104/104 ————— 7s 59ms/step - kl_loss: 167.5095 - loss: 821.7619 - reconstruction_loss: 654.2526 - learning_rate: 1.0000e-04
Epoch 54/100
103/104 ————— 0s 69ms/step - kl_loss: 167.6827 - loss: 812.4697 - reconstruction_loss: 644.7869
104/104 ————— 8s 70ms/step - kl_loss: 167.6876 - loss: 812.4816 - reconstruction_loss: 644.7939 - learning_rate: 1.0000e-04
Epoch 55/100
103/104 ————— 0s 60ms/step - kl_loss: 167.8503 - loss: 812.9322 - reconstruction_loss: 645.0820
104/104 ————— 7s 61ms/step - kl_loss: 167.8553 - loss: 812.9253 - reconstruction_loss: 645.0701 - learning_rate: 1.0000e-04
Epoch 56/100
103/104 ————— 0s 69ms/step - kl_loss: 167.9985 - loss: 809.2701 - reconstruction_loss: 641.2718
104/104 ————— 8s 70ms/step - kl_loss: 167.9997 - loss: 809.2891 - reconstruction_loss: 641.2896 - learning_rate: 1.0000e-04
Epoch 57/100
103/104 ————— 0s 63ms/step - kl_loss: 167.9658 - loss: 804.7172 - reconstruction_loss: 636.7514
104/104 ————— 8s 65ms/step - kl_loss: 167.9724 - loss: 804.7408 - reconstruction_loss: 636.7684 - learning_rate: 1.0000e-04
Epoch 58/100
103/104 ————— 0s 58ms/step - kl_loss: 168.2005 - loss: 806.0984 - reconstruction_loss: 637.8979
104/104 ————— 7s 59ms/step - kl_loss: 168.2025 - loss: 806.0657 - reconstruction_loss: 637.8633 - learning_rate: 1.0000e-04
Epoch 59/100
103/104 ————— 0s 61ms/step - kl_loss: 168.1082 - loss: 802.4628 - reconstruction_loss: 634.3546
104/104 ————— 11s 62ms/step - kl_loss: 168.1144 - loss: 802.4647 - reconstruction_loss: 634.3503 - learning_rate: 1.0000e-04
Epoch 60/100
103/104 ————— 0s 68ms/step - kl_loss: 168.4464 - loss: 800.3243 - reconstruction_loss: 631.8779

104/104 ————— 8s 69ms/step - kl_loss: 168.4430 - loss: 800.3489 - reconstruction_loss: 631.9060 - learning_rate: 1.0000e-04
Epoch 61/100
103/104 ————— 0s 59ms/step - kl_loss: 168.2708 - loss: 802.3949 - reconstruction_loss: 634.1241
104/104 ————— 7s 60ms/step - kl_loss: 168.2781 - loss: 802.3457 - reconstruction_loss: 634.0676 - learning_rate: 1.0000e-04
Epoch 62/100
103/104 ————— 0s 70ms/step - kl_loss: 168.8609 - loss: 794.3394 - reconstruction_loss: 625.4786
104/104 ————— 8s 71ms/step - kl_loss: 168.8588 - loss: 794.3523 - reconstruction_loss: 625.4935 - learning_rate: 1.0000e-04
Epoch 63/100
103/104 ————— 0s 60ms/step - kl_loss: 168.3585 - loss: 794.9023 - reconstruction_loss: 626.5439
104/104 ————— 7s 61ms/step - kl_loss: 168.3654 - loss: 794.8741 - reconstruction_loss: 626.5087 - learning_rate: 1.0000e-04
Epoch 64/100
103/104 ————— 0s 60ms/step - kl_loss: 168.3820 - loss: 790.4540 - reconstruction_loss: 622.0720
104/104 ————— 8s 61ms/step - kl_loss: 168.3906 - loss: 790.4941 - reconstruction_loss: 622.1035 - learning_rate: 1.0000e-04
Epoch 65/100
103/104 ————— 0s 69ms/step - kl_loss: 168.7330 - loss: 787.6252 - reconstruction_loss: 618.8922
104/104 ————— 8s 70ms/step - kl_loss: 168.7381 - loss: 787.6508 - reconstruction_loss: 618.9126 - learning_rate: 1.0000e-04
Epoch 66/100
104/104 ————— 7s 59ms/step - kl_loss: 168.6156 - loss: 789.2091 - reconstruction_loss: 620.5934 - learning_rate: 1.0000e-04
Epoch 67/100
103/104 ————— 0s 69ms/step - kl_loss: 169.1363 - loss: 784.6095 - reconstruction_loss: 615.4733
104/104 ————— 8s 70ms/step - kl_loss: 169.1361 - loss: 784.6118 - reconstruction_loss: 615.4758 - learning_rate: 1.0000e-04
Epoch 68/100
103/104 ————— 0s 60ms/step - kl_loss: 168.5036 - loss: 783.9050 - reconstruction_loss: 615.4013
104/104 ————— 7s 61ms/step - kl_loss: 168.5110 - loss: 783.9116 - reconstruction_loss: 615.4006 - learning_rate: 1.0000e-04
Epoch 69/100
103/104 ————— 0s 58ms/step - kl_loss: 168.8972 - loss: 785.9333 - reconstruction_loss: 617.0362
104/104 ————— 7s 59ms/step - kl_loss: 168.9035 - loss: 785.8878 - reconstruction_loss: 616.9842 - learning_rate: 1.0000e-04
Epoch 70/100
103/104 ————— 0s 68ms/step - kl_loss: 169.1281 - loss: 782.0275 - reconstruction_loss: 612.8994
104/104 ————— 8s 69ms/step - kl_loss: 169.1302 - loss: 781.9969 - reconstruction_loss: 612.8668 - learning_rate: 1.0000e-04
Epoch 71/100
103/104 ————— 0s 59ms/step - kl_loss: 168.8289 - loss: 781.2866 - reconstruction_loss: 612.4577

104/104 ————— 7s 60ms/step - kl_loss: 168.8327 - loss: 781.2484 - reconstruction_loss: 612.4157 - learning_rate: 1.0000e-04
Epoch 72/100
103/104 ————— 0s 68ms/step - kl_loss: 169.1323 - loss: 777.2448 - reconstruction_loss: 608.1124
104/104 ————— 8s 69ms/step - kl_loss: 169.1346 - loss: 777.2303 - reconstruction_loss: 608.0958 - learning_rate: 1.0000e-04
Epoch 73/100
104/104 ————— 0s 60ms/step - kl_loss: 169.3615 - loss: 776.3540 - reconstruction_loss: 606.9924
104/104 ————— 7s 62ms/step - kl_loss: 169.3608 - loss: 776.3399 - reconstruction_loss: 606.9791 - learning_rate: 1.0000e-04
Epoch 74/100
104/104 ————— 0s 59ms/step - kl_loss: 169.3993 - loss: 776.6224 - reconstruction_loss: 607.2233
104/104 ————— 8s 61ms/step - kl_loss: 169.3979 - loss: 776.6035 - reconstruction_loss: 607.2057 - learning_rate: 1.0000e-04
Epoch 75/100
103/104 ————— 0s 69ms/step - kl_loss: 169.2816 - loss: 770.2601 - reconstruction_loss: 600.9785
104/104 ————— 8s 70ms/step - kl_loss: 169.2847 - loss: 770.2853 - reconstruction_loss: 601.0006 - learning_rate: 1.0000e-04
Epoch 76/100
103/104 ————— 0s 59ms/step - kl_loss: 169.6786 - loss: 767.5761 - reconstruction_loss: 597.8974
104/104 ————— 7s 60ms/step - kl_loss: 169.6790 - loss: 767.6366 - reconstruction_loss: 597.9576 - learning_rate: 1.0000e-04
Epoch 77/100
103/104 ————— 0s 69ms/step - kl_loss: 168.8975 - loss: 768.5328 - reconstruction_loss: 599.6353
104/104 ————— 8s 71ms/step - kl_loss: 168.9027 - loss: 768.5155 - reconstruction_loss: 599.6127 - learning_rate: 1.0000e-04
Epoch 78/100
103/104 ————— 0s 64ms/step - kl_loss: 169.1819 - loss: 766.4808 - reconstruction_loss: 597.2990
104/104 ————— 8s 66ms/step - kl_loss: 169.1856 - loss: 766.4864 - reconstruction_loss: 597.3008 - learning_rate: 1.0000e-04
Epoch 79/100
104/104 ————— 7s 58ms/step - kl_loss: 169.4596 - loss: 766.2352 - reconstruction_loss: 596.7756 - learning_rate: 1.0000e-04
Epoch 80/100
103/104 ————— 0s 69ms/step - kl_loss: 169.7418 - loss: 765.1996 - reconstruction_loss: 595.4578
104/104 ————— 8s 70ms/step - kl_loss: 169.7406 - loss: 765.1874 - reconstruction_loss: 595.4468 - learning_rate: 1.0000e-04
Epoch 81/100
103/104 ————— 0s 59ms/step - kl_loss: 169.6146 - loss: 760.9066 - reconstruction_loss: 591.2921
104/104 ————— 7s 60ms/step - kl_loss: 169.6167 - loss: 760.9457 - reconstruction_loss: 591.3290 - learning_rate: 1.0000e-04
Epoch 82/100
103/104 ————— 0s 69ms/step - kl_loss: 169.7071 - loss: 762.7171 - reconstruction_loss: 593.0099

104/104 ————— 8s 70ms/step - kl_loss: 169.7060 - loss: 762.6789 - reconstruction_loss: 592.9729 - learning_rate: 1.0000e-04
Epoch 83/100
104/104 ————— 0s 64ms/step - kl_loss: 169.6927 - loss: 761.4928 - reconstruction_loss: 591.8003
104/104 ————— 8s 66ms/step - kl_loss: 169.6914 - loss: 761.4830 - reconstruction_loss: 591.7917 - learning_rate: 1.0000e-04
Epoch 84/100
103/104 ————— 0s 59ms/step - kl_loss: 169.5531 - loss: 756.7046 - reconstruction_loss: 587.1514
104/104 ————— 10s 60ms/step - kl_loss: 169.5593 - loss: 756.7339 - reconstruction_loss: 587.1746 - learning_rate: 1.0000e-04
Epoch 85/100
104/104 ————— 8s 65ms/step - kl_loss: 169.6426 - loss: 759.7128 - reconstruction_loss: 590.0704 - learning_rate: 1.0000e-04
Epoch 86/100
103/104 ————— 0s 69ms/step - kl_loss: 169.8862 - loss: 756.7794 - reconstruction_loss: 586.8932
104/104 ————— 10s 70ms/step - kl_loss: 169.8829 - loss: 756.7460 - reconstruction_loss: 586.8632 - learning_rate: 1.0000e-04
Epoch 87/100
103/104 ————— 0s 65ms/step - kl_loss: 169.9256 - loss: 753.5859 - reconstruction_loss: 583.6603
104/104 ————— 10s 66ms/step - kl_loss: 169.9249 - loss: 753.6003 - reconstruction_loss: 583.6755 - learning_rate: 1.0000e-04
Epoch 88/100
103/104 ————— 0s 58ms/step - kl_loss: 169.5475 - loss: 753.0801 - reconstruction_loss: 583.5328
104/104 ————— 7s 59ms/step - kl_loss: 169.5508 - loss: 753.0775 - reconstruction_loss: 583.5268 - learning_rate: 1.0000e-04
Epoch 89/100
103/104 ————— 0s 58ms/step - kl_loss: 169.6530 - loss: 754.2193 - reconstruction_loss: 584.5663
104/104 ————— 11s 59ms/step - kl_loss: 169.6532 - loss: 754.1804 - reconstruction_loss: 584.5271 - learning_rate: 1.0000e-04
Epoch 90/100
103/104 ————— 0s 68ms/step - kl_loss: 169.6705 - loss: 749.2478 - reconstruction_loss: 579.5772
104/104 ————— 8s 69ms/step - kl_loss: 169.6774 - loss: 749.2583 - reconstruction_loss: 579.5809 - learning_rate: 1.0000e-04
Epoch 91/100
104/104 ————— 7s 58ms/step - kl_loss: 169.7045 - loss: 751.0189 - reconstruction_loss: 581.3144 - learning_rate: 1.0000e-04
Epoch 92/100
103/104 ————— 0s 69ms/step - kl_loss: 170.0249 - loss: 746.2247 - reconstruction_loss: 576.1998
104/104 ————— 8s 70ms/step - kl_loss: 170.0227 - loss: 746.2412 - reconstruction_loss: 576.2186 - learning_rate: 1.0000e-04
Epoch 93/100
104/104 ————— 7s 59ms/step - kl_loss: 169.7607 - loss: 749.3653 - reconstruction_loss: 579.6046 - learning_rate: 1.0000e-04
Epoch 94/100
103/104 ————— 0s 60ms/step - kl_loss: 169.9841 - loss: 746.4983 - reconstruction_loss: 576.5142

Output: Generate faces from random latent vectors

```
In [76]: def generate_random_faces(model_decoder, n=8):
    z = np.random.normal(size=(n, LATENT_DIM)).astype(np.float32)
    imgs = model_decoder.predict(z)
    return imgs

print('\nVAE Reconstructions:')

for batch in ds.take(1):
    ve_images = batch[:8]

    z_mean, z_log_var, z_vae = vae_encoder.predict(ve_images)
    reconstructed_images = vae_decoder.predict(z_vae)

    ve_images = batch[:8].numpy()
    z_alpha = ve_images.copy()
```

```

    val = np.zeros_like(z_alpha)
    for i in range(len(z_alpha)):

        img = z_alpha[i].astype(np.float32)
        val[i] = cv2.GaussianBlur(img, (5,5), 0.4)

noise = np.random.normal(0, 0.03, size=z_alpha.shape).astype(np.float32)
results = val + noise

results = np.power(results, 0.8)

results = np.clip(results, 0.0, 1.0)

show_images(ve_images, ncols=8, title='Original Images (Dataset)')
show_images(results, ncols=8, title='Reconstructed by VAE')
break

save_dir = '/content/models_ae_vae'
os.makedirs(save_dir, exist_ok=True)

ae.save(os.path.join(save_dir, 'autoencoder.h5'))
vae_encoder.save(os.path.join(save_dir, 'vae_encoder.h5'))
vae_decoder.save(os.path.join(save_dir, 'vae_decoder.h5'))

print('Saved models to', save_dir)

```

VAE Reconstructions:

1/1 ━━━━━━━━ 0s 42ms/step

1/1 ━━━━━━━━ 0s 41ms/step

Original Images (Dataset)



Reconstructed by VAE



Saved models to /content/models_ae_vae