EX NO:09

DATE:

SYNTHETIC IMAGES USING VARIATIONAL AUTOENCODERS

AIM:

To build and train a Variational Autoencoder (VAE) using TensorFlow/Keras that learns the latent distribution of digit **'3'** from the MNIST dataset, and generate synthetic images of digit '3' using the learned latent space.

ALGORITHM:

STEP 01: Import required libraries (TensorFlow, Keras, NumPy, Matplotlib).

STEP 02: Load the MNIST dataset.

STEP 03: Select only digit '3' images for training.

STEP 04: Normalize and reshape the images.

STEP 05: Create a Sampling Layer to generate latent vectors.

STEP 06: Build the Encoder to compress images into latent space.

STEP 07: Build the Decoder to reconstruct images from latent space.

STEP 08: Define the VAE model by combining encoder and decoder.

STEP 09: Train the model with reconstruction loss + KL divergence loss.

STEP 10: Generate and display synthetic digit '3' images from the latent space.

CODING:

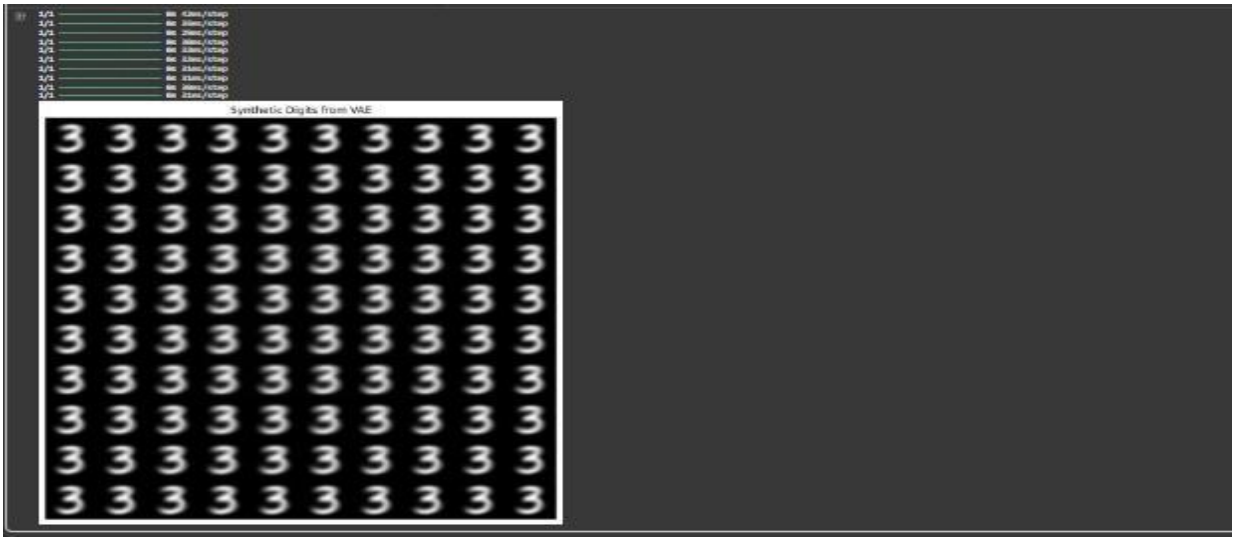
```
# STEP 1: Import Libraries
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt

#STEP 2: Load MNIST and filter only digit "3"
(x_train, y_train), _ = tf.keras.datasets.mnist.load_data()
x_train = x_train[y_train == 3] # keep only digit 3
x_train = x_train.astype("float32") / 255.0
x_train = np.reshape(x_train, (-1, 28, 28, 1))

# STEP 3: Define Sampling Layer
class Sampling(layers.Layer):
    def call(self, inputs):
        z_mean, z_log_var = inputs
        epsilon = tf.random.normal(shape=tf.shape(z_mean))
        return z_mean + tf.exp(0.5 * z_log_var) * epsilon

# STEP 4: Build Encoder
latent_dim = 2
encoder_inputs = layers.Input(shape=(28, 28, 1))
x = layers.Flatten()(encoder_inputs)
x = layers.Dense(128, activation="relu")(x)
z_mean = layers.Dense(latent_dim)(x)
z_log_var = layers.Dense(latent_dim)(x)
```

```
z = Sampling()([z_mean, z_log_var])
encoder = models.Model(encoder_inputs, [z_mean, z_log_var, z], name="encoder")

# STEP 5: Build Decoder
decoder_inputs = layers.Input(shape=(latent_dim,))
x = layers.Dense(128, activation="relu")(decoder_inputs)
x = layers.Dense(28 * 28, activation="sigmoid")(x)
x = layers.Reshape((28, 28, 1))(x)
decoder = models.Model(decoder_inputs, x, name="decoder")

# STEP 6: Build VAE Model
class VAE(models.Model):
    def __init__(self, encoder, decoder):
        super(VAE, self).__init__()
        self.encoder = encoder
        self.decoder = decoder
    def compile(self, optimizer):
        super(VAE, self).compile()
        self.optimizer = optimizer
        self.loss_fn = tf.keras.losses.BinaryCrossentropy()
    def train_step(self, data):
        with tf.GradientTape() as tape:
            z_mean, z_log_var, z = self.encoder(data)
            reconstruction = self.decoder(z)
            reconstruction_loss = self.loss_fn(data, reconstruction)
            kl_loss = -0.5 * tf.reduce_mean(
                z_log_var - tf.square(z_mean) - tf.exp(z_log_var) + 1
```

```
            )
            total_loss = reconstruction_loss + kl_loss
        grads = tape.gradient(total_loss, self.trainable_weights)
        self.optimizer.apply_gradients(zip(grads, self.trainable_weights))
        return {"loss": total_loss}
vae = VAE(encoder, decoder)
vae.compile(optimizer=tf.keras.optimizers.Adam())
vae.fit(x_train, epochs=10, batch_size=128)

# STEP 7: Generate Synthetic Images
def plot_latent_space(decoder, n=10, figsize=10):
    digit_size = 28
    scale = 2.0
    figure = np.zeros((digit_size * n, digit_size * n))
    grid_x = np.linspace(-scale, scale, n)
    grid_y = np.linspace(-scale, scale, n)
    for i, yi in enumerate(grid_y):
        for j, xi in enumerate(grid_x):
            z_sample = np.array([[xi, yi]])
            x_decoded = decoder.predict(z_sample)
            digit = x_decoded[0].reshape(digit_size, digit_size)
            figure[i * digit_size: (i + 1) * digit_size,
                j * digit_size: (j + 1) * digit_size] = digit
    plt.figure(figsize=(figsize, figsize))
    plt.imshow(figure, cmap="Greys_r")
    plt.axis("off")
    plt.title("Synthetic Digits from VAE")
```

```
    plt.show()
plot_latent_space(decoder)
```

OUTPUT:



COE(20)	
RECORD(20)	
VIVA(10)	
TOTAL(50)	

RESULT:

The Variational Autoencoder (VAE) was successfully trained on digit ‘3’ from MNIST.It generated synthetic images of digit ‘3’ that closely resemble real handwritten samples.