**Logo Reconstruction using Variational Autoencoders (VAE)**

In this exercise you are tasked to do **image reconstruction** of some popular logo. You must use **Variational Autoencoders** model to do this task. Follow the instructions below for more details. Good luck!

* **Load and Split Dataset**
  + **Load** **Dataset**

In this section, you are required to:

* + - **Assign** the dataset path.
    - **Load** the image from the given dataset into an array using RGB as the color mode.
    - **Randomly** shuffle the dataset.
    - **Resize** or use target size of 100 x 100 for better result
  + **Split** **Dataset**

You are tasked to **split** the given dataset into **2 main parts** which are training and testing sets with the **last 10 images** as the **testing sets** and the **rest** as **training sets**.

* **Data Preprocessing**

You are tasked with **reshaping** the given dataset, which includes the training and testing sets, by converting the data type to float32. Following this, you will need to **normalize** the dataset, comprising the training and testing sets, by dividing the values of all image pixels by 255.

* **Variational Autoencoders Architecture (Model Architecture Visualization, Initialization, and Configuration)**
  + **Model Visualization**

A diagram of a flowchart

Description automatically generated

**Figure 1. Model Visualization**

* + **Model Initialization and Configuration – Encoder**

**Input Layer:** This takes the data (like an image) as input.

**Hidden Layers:** One or more layers that transform the input data. These can be fully connected layers, convolutional layers, or any other type of layer suitable for the data type.

**Output Layer:** Produces two vectors for each input data point: a mean vector and a log variance vector .

These vectors are used to parameterize the Gaussian distribution from which we’ll sample the latent variable ‘z’.

* + **Model Initialization and Configuration – Latent Space**

This is a lower-dimensional space where we encode our data. A point ‘z’ in this space is sampled from the Gaussian distribution parameterized by and from the encoder. The **reparameterization trick** is used here to make this sampling process differentiable. Instead of directly sampling from the distribution, we sample from a standard normal distribution and then scale and shift using µ and σ. The **formula** for the **reparameterization** **trick** is , where μ is the mean vector produced by the encoder, σ is the standard deviation, and ϵ is a random noise sampled from a standard normal distribution.

* + **Model Initialization and Configuration – Decoder**

**Input Layer:** Takes the sampled ‘z’ from the latent space.

**Hidden Layers:** One or more layers that transform the latent variable back towards the original data’s dimensionality.

**Output Layer:** Produces the reconstructed data, ideally as close to the original input data as possible.

* **Model Training**

You are tasked with **training** **the model** that you previously constructed. **Utilize** the Adam optimizer and **select** the appropriate loss function based on the specifics of your case. **Monitor** the performance using **reconstruction loss** and **KL Divergence** as the metric. Ensure that the training process spans a minimum of **25 epochs**. You have the flexibility to set the batch size and learning rate for the optimizer according to your judgment.

A table with numbers and a loss

Description automatically generated with medium confidence

**Figure 2. Model Training**

* **Predicting Data Using Created Model and Model Evaluation**

You are tasked with **evaluating** the testing sets using the model that you previously constructed. **Display** both the **original and the reconstructed images**, **showcasing 10 images** **each**.

**A group of logos with different colors

Description automatically generated with medium confidence**

**Figure 3. Original Images and Reconstructed Images**