**Step 1: Global Probabilistic Guarantees with VAEs**

**Objective**: Design and implement a module using Variational Autoencoders (VAEs) to generate a diverse set of inputs that represent the entire data distribution for testing.

Design:

1. **VAE Model Architecture**: The architecture will consist of an encoder and a decoder. The encoder compresses the input into a latent space representation, and the decoder reconstructs the input from this latent space. The model will be trained on your dataset of interest.
2. **Input Sampling**: After training the VAE, we use the decoder part to sample new inputs by generating points in the latent space and decoding them into inputs. These generated inputs should capture the variability within the original dataset, allowing us to test the model with unseen but plausible data.

Implementation:

* **VAE Architecture Setup**: Define the encoder and decoder architectures using a deep learning framework like TensorFlow or PyTorch.
* **Training the VAE**: Use your dataset to train the VAE, ensuring that the loss function includes both the reconstruction loss and the KL divergence to encourage a well-formed latent space.
* **Sampling New Inputs**: Implement a function to sample points from the latent space and decode them to generate new test inputs.

**Global vs. Local Probabilistic Guarantees**

* **Local Probabilistic Guarantees** refer to the assurance that a deep learning model's predictions are robust for a specific input or a small neighborhood around that input. It is about ensuring the model's performance on slight variations of a particular input, which can be tested through methods like adversarial testing.
* **Global Probabilistic Guarantees**, on the other hand, aim to provide assurance over the entire input space or a significant portion of it. It's about the model's overall robustness across diverse input scenarios, not just close neighbors of seen examples. This is where generative models like VAEs come into play.

**How VAEs Contribute to Global Probabilistic Guarantees**

VAEs can generate new data points that are representative of the entire data distribution seen during training. By sampling from the latent space, VAEs can produce inputs that cover a wide variety of scenarios, including edge cases that might not be present in the original training dataset. Here’s how this ties back to the concept of global probabilistic guarantees:

1. **Diverse Input Generation**: By efficiently exploring the latent space, VAEs can generate a diverse set of inputs that are plausible based on the training distribution. This helps in testing the model against a wide variety of inputs, thus moving towards a global guarantee.
2. **Sampling Strategy**: The latent space of a well-trained VAE should encode meaningful variations in the data. By strategically sampling from this space (e.g., covering different regions of the latent space evenly), we can ensure that the generated test cases are not just random but systematically explore potential weaknesses of the model across its global input space.
3. **Coverage Measurement**: To quantify global probabilistic guarantees, one approach is to measure the coverage of the generated test cases across the input space. This could involve metrics that assess how well the sampled points from the VAE's latent space represent the diversity of the entire input distribution.

**Implementing Global Probabilistic Guarantees in Code**

In the provided VAE implementation, global probabilistic guarantees are implicitly addressed through the design of the VAE itself and how we choose to sample from it. Here’s what can be explicitly done in code to align with the concept:

* **Uniform Sampling**: Ensure that the sampling from the latent space is done in a manner that covers it uniformly or according to a desired distribution that matches the diversity of the data. This can be as simple as sampling from a standard normal distribution if the VAE is trained appropriately, or it might involve more sophisticated sampling strategies to ensure coverage.
* **Latent Space Exploration**: Implement functions to systematically explore the latent space, for example, by generating samples that traverse along different dimensions of the latent space to see how changes in each dimension affect the output.
* **Evaluation Metrics**: After generating inputs using the VAE, use them to test the pretrained model and evaluate the model’s performance across these inputs. Metrics could include not just accuracy but also measures of confidence distribution, error types, and error rates across different regions of the input space.

To reflect this in the context of the CIFAR-10 and ResNet example, after generating new images using the VAE, you would:

* **Classify Generated Images**: Use the pretrained ResNet model to classify these images, observing the distribution of predictions to assess model robustness.
* **Analyze Misclassifications**: Identify and categorize errors or misclassifications by the ResNet model to understand if there are specific patterns or regions of the input space where the model struggles.

This approach not only tests the model in a more comprehensive manner but also aligns with the goal of achieving global probabilistic guarantees by ensuring the model is robust across a wide array of input scenarios generated from the learned data distribution.