**1. Main Tasks of Autoencoders**

Autoencoders (AEs) are used for:

✅ **Dimensionality Reduction** → Similar to PCA but non-linear (e.g., feature extraction).  
✅ **Denoising** → Removing noise from images, audio, or text (e.g., Denoising Autoencoders).  
✅ **Anomaly Detection** → Detecting fraud or defects by learning normal patterns.  
✅ **Data Compression** → Encoding high-dimensional data into a compact representation.  
✅ **Generative Modeling** → Creating new data similar to input (e.g., Variational Autoencoders).

**2. How Autoencoders Help When Labeled Data is Scarce?**

📌 **Solution: Unsupervised Pretraining with Autoencoders**

1. **Train an autoencoder** on the large **unlabeled dataset**.
2. **Extract latent features** from the bottleneck layer.
3. **Train a classifier** (e.g., logistic regression, MLP) using the labeled subset.

✅ **Benefits:**

* Reduces the need for labeled data.
* Learns meaningful representations, improving classification performance.
* Works well in **semi-supervised learning** settings.

**3. Is Perfect Reconstruction a Sign of a Good Autoencoder?**

❌ **Not necessarily.**

🚨 **Potential Issues:**

* **Overfitting**: The AE memorizes data instead of learning meaningful representations.
* **Identity Mapping**: If the encoder-decoder just copies inputs, it’s useless.

📌 **Better Evaluation Metrics:**

* **Reconstruction Error**: Compare input vs. output using **MSE or SSIM**.
* **Latent Space Quality**: Check clustering of features in the latent space.
* **Generalization**: See if the AE can reconstruct unseen data well.

**4. Undercomplete vs. Overcomplete Autoencoders**

📌 **Undercomplete AE**:

* **Latent space is smaller** than the input.
* **Forces learning of key patterns** (useful for feature extraction).
* **Risk**: If too compressed, it might lose useful information.

📌 **Overcomplete AE**:

* **Latent space is larger** than input.
* **More capacity but risk of memorization** (loses generalization).
* **Risk**: If not regularized, it can just copy inputs instead of learning.

⚡ **Solution?**

* Use **dropout, weight regularization, or sparsity constraints** to prevent overfitting.

**5. Tying Weights in a Stacked Autoencoder**

📌 **Tied Weights**:

* Instead of learning separate weights for encoding & decoding, we **reuse them** (i.e., decoder weights = encoder weights transposed).
* **Reduces the number of parameters** → Prevents overfitting.

✅ **Benefits:**

* **Improves generalization**.
* **Speeds up training**.
* **Encourages symmetry in encoding-decoding**.

**6. What is a Generative Model? Example of a Generative Autoencoder?**

🧠 **Generative Model** = A model that learns the **data distribution** and generates **new samples**.

📌 **Example: Variational Autoencoder (VAE)**

* Instead of deterministic encoding, it learns a **probability distribution** in latent space.
* Uses **KL divergence loss** to encourage meaningful latent representations.
* **Great for** image generation, text synthesis, and style transfer.

**7. What is a GAN? Where Do GANs Shine?**

🚀 **Generative Adversarial Network (GAN)** consists of:

1. **Generator**: Creates fake samples.
2. **Discriminator**: Distinguishes real vs. fake samples.
3. **Training = Minimax Game**: Generator tries to fool the discriminator, improving generation quality.

📌 **Best Uses for GANs**:  
✅ **Image Synthesis** → DeepFake, AI art (e.g., DALL·E).  
✅ **Super-Resolution** → Upscaling low-res images.  
✅ **Style Transfer** → Making photos look like paintings.  
✅ **Data Augmentation** → Generating synthetic training samples.

**8. Main Challenges in Training GANs**

🚨 **Common Issues & Fixes:**

❌ **Mode Collapse** → Generator produces limited variations.  
💡 **Fix**: Use **diversity-promoting losses** (e.g., minibatch discrimination).

❌ **Vanishing Gradients** → Discriminator becomes too strong.  
💡 **Fix**: Use **Wasserstein Loss (WGANs)** for stable training.

❌ **Unstable Training** → Oscillations in loss values.  
💡 **Fix**: Use **batch normalization** and **gradient clipping**.