### ****Q1. What is a loss function? How is it different from a cost function?****

👉 **Loss function**: Error for a single training example.  
👉 **Cost function**: Average of loss over the dataset.  
(Though in practice, people use them interchangeably.)

**Q2. Explain the difference between MSE, MAE, and Huber loss.**

* **MSE (Mean Squared Error)**: Penalizes large errors heavily (quadratic). Smooth gradients, but sensitive to outliers.
* **MAE (Mean Absolute Error)**: Linear penalty → more robust to outliers. But gradient is constant, which slows learning.
* **Huber Loss**: Quadratic near 0, linear for large errors → combines benefits of both.

### ****Q3. Why is cross-entropy loss preferred over MSE in classification?****

👉 MSE gives slow convergence because it doesn’t align well with softmax outputs (gradients get very small).  
👉 Cross-entropy directly measures the distance between probability distributions (predicted vs. true) → faster and more stable training.

**Q4. What is categorical cross-entropy vs. binary cross-entropy?**

* **Binary Cross-Entropy (BCE)**: For 2-class problems (output ∈ {0,1}).

L=−[ylog (p)+(1−y)log (1−p)]]

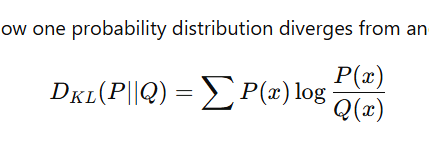
* **Categorical Cross-Entropy (CCE)**: For multi-class (softmax output).

L=−∑iyilog (pi)

If labels are one-hot → categorical; if labels are integers → sparse categorical.

**Q5. What is KL divergence loss? When is it used?**

👉 KL divergence measures how one probability distribution diverges from another:



Used in:

* Variational Autoencoders (VAEs).
* Knowledge Distillation (teacher vs. student network).

### ****Q6. What is focal loss and why is it useful?****

👉 Focal loss modifies cross-entropy by adding a factor that downweights well-classified examples → focuses on hard/misclassified examples.  
Used in object detection (e.g., RetinaNet) for **class imbalance**.

**Q7. How do you choose a loss function for regression vs. classification vs. ranking?**

* **Regression** → MSE, MAE, Huber.
* **Classification** → Cross-entropy, Focal loss.
* **Ranking** → Hinge loss, Contrastive loss, Triplet loss.

**Q9. What are custom loss functions? Can you give an example?**

👉 Custom losses encode business-specific goals. Example:

* In marketing: cost-sensitive loss → weight conversions higher than impressions.
* In imbalanced datasets: weighted BCE.  
  Example (weighted BCE):

L=−[w1 ylog(p) + w0 (1−y) log(1−p)]

**Q11. Why can’t accuracy be used as a loss function?**

👉 Accuracy is non-differentiable (step function). Optimizers like SGD need gradients, so accuracy can’t guide weight updates.

**Q12. What happens if you use the wrong loss function?**

👉 Model may converge to a suboptimal solution. Examples:

* Using MSE for classification → slow convergence.
* Using cross-entropy for regression → nonsensical outputs.

**Q13. How do ranking losses (contrastive / triplet) work?**

* **Contrastive loss**: Encourages similar pairs to have small distance, dissimilar pairs to be apart.
* **Triplet loss**: Minimizes distance(anchor, positive) – distance(anchor, negative) + margin.  
  👉 Used in face recognition, recommendation systems.

**Q14. How do you handle imbalanced datasets with loss functions?**

* Weighted cross-entropy.
* Focal loss.
* Oversampling + loss adjustments.  
  👉 Example: In medical imaging, false negatives are more costly → increase their weight in loss.

**Q15. What’s the intuition behind label smoothing in loss functions?**

👉 Instead of hard labels (0 or 1), assign soft targets (e.g., 0.9 for true class, 0.1 spread across others).  
✅ Prevents overconfidence, improves generalization (used in Transformers).

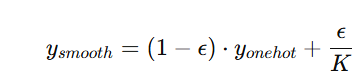
**🔹 What is Label Smoothing?**

👉 **Label smoothing** is a regularization technique where instead of using hard one-hot labels (e.g., [0,0,1,0]), we **soften the labels** by distributing a small amount of probability mass to all other classes.

Example (4-class problem, true class = 3):

* **Without smoothing (one-hot):** [0, 0, 1, 0]
* **With smoothing (ε=0.1):** [0.033, 0.033, 0.9, 0.033]

Formula:



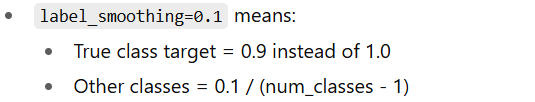
where **K = number of classes**, and **ε = smoothing factor** (typically 0.05–0.2).

**🔹 Where is Label Smoothing Used?**

1. **Classification tasks** with softmax output (image classification, NLP).
2. **Transformers (NLP/CV)** → used in **attention-based models** like BERT, GPT training.
3. **Image classification benchmarks** (ResNet, EfficientNet, Vision Transformer).
4. **Distillation** → when training a student from a teacher network.

**🔹 Why Use Label Smoothing?**

1. **Prevents overconfidence:**
   * With one-hot labels, the model can become too confident (p=0.999 for one class).
   * Overconfidence hurts generalization → poor performance on unseen data.
   * Label smoothing encourages the model to keep probabilities softer.
2. **Improves generalization:**
   * Reduces variance, helps avoid overfitting.
   * Especially useful in small datasets or imbalanced datasets.
3. **Stabilizes training:**
   * Avoids **infinite loss** when the model predicts 0 probability for the correct class.
   * Helps prevent gradient spikes.
4. **Better calibrated probabilities:**
   * Outputs are more interpretable (probabilities closer to true likelihood).



**🔹 Why this helps**

* Prevents overconfidence.
* Regularizes training.
* Improves calibration of predicted probabilities.