# AUTOGRAD - SHORT NOTES

## SUMMARY

Here are the concise revision notes for Autograd:

Definitions

1. Autograd: System for automatic differentiation, computing gradients of a function with respect to its inputs.
2. Gradient: Measure of how much each parameter contributes to the loss function.
3. Automatic Differentiation: Technique for computing gradients without explicitly defining the derivative.
4. Tensor: Multi-dimensional array of numerical values.
5. Node: Basic building block of a computation graph, representing a single operation or function.
6. Computation Graph: Directed acyclic graph (DAG) representing the flow of data through a neural network.

Key Concepts

1. Forward Pass: Computing output of a neural network given its inputs, building a computation graph.
2. Backward Pass: Computing gradients of the loss function with respect to the model's parameters, traversing the computation graph in reverse.
3. Gradient Accumulation: Accumulating gradients of the loss function with respect to each parameter.

Syntax/Code Examples

1. PyTorch: `x = torch.tensor([1., 2., 3.]); y = x2; y.backward()`
2. TensorFlow: `x = tf.constant([1., 2., 3.]); y = x2; gradients = tf.gradients(y, x)`

Common Mistakes

1. Forgetting to compute gradients during the backward pass.
2. Not accumulating gradients correctly.

Best Practices / Tips

1. Use Autograd for automatic differentiation in neural networks.
2. Build computation graphs to visualize data flow.
3. Use gradient accumulation to compute gradients recursively.

Applications

1. Neural Network Training: Computing gradients for training neural networks.
2. Deep Learning Frameworks: Key component of PyTorch, TensorFlow, and Keras.
3. Automatic Differentiation: Used in scientific computing, optimization, and signal processing.

These notes cover all essential points, are concise, and easy to review. They are perfect for quick last-minute interview revision.