Deep Learning Assignment Report

1. Problem Understanding
- Task: Build a Convolutional Neural Network (CNN) for image classification using the CIFAR-10 dataset.

2. Model Design
- Architecture: 3 convolutional layers, 2 fully connected layers.
- Features: ReLU activation, Batch Normalization, and Dropout.

3. Results and Discussion
- Training/Validation Curves: Included.
- Test Accuracy: <Placeholder for test accuracy>.
- Confusion Matrix: Attached.

1. Activation Functions

(a) Sigmoid:

- **Definition**: A mathematical function that maps input values to a range between 0 and 1 using the formula:

$$\boldsymbol{f(x) = \frac{1}{1+e^{-x}}}$$

- **Advantages**: Smooth gradient; interpretable output for probabilities.
- **Limitations**: Vanishing gradient problem; slow convergence; not zero-centered.
- **Use Case**: Logistic regression, binary classification problems.

(b) ReLU (Rectified Linear Unit):

Definition: Outputs the input directly if positive, otherwise output zero:

$$f(x) = \max(0, x)$$

- **Advantages**: Computationally efficient; mitigates vanishing gradient problems; promotes sparse activations.

- **Limitations**: Can suffer from the "dead neurons" issue.
- **Use Case**: Standard for most deep learning tasks, especially CNNs.

(c) Tanh:

- **Definition**: A hyperbolic tangent function that outputs values between -1 and 1:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- **Advantages**: Zero-centered; better for centered data than Sigmoid.
- **Limitations**: Suffers from vanishing gradient issues.
- **Use Case**: Older neural network models; use has declined in favor of ReLU.

(d) Leaky ReLU:

- **Definition**: Allows small gradients for negative inputs by modifying the ReLU formula:

$$f(x) = \left\{ \begin{matrix} x, & x > 0 \\ ax, & x \leq 0 \end{matrix} \right\}$$

- **Advantages**: Solves the dead neuron issue in ReLU.
- **Limitations**: Slightly more computationally expensive.
- **Use Case**: Alternative to ReLU in deeper networks.

2. Optimization Algorithms

Here's the explanation for your report:

(a) SGD (Stochastic Gradient Descent):

- **Description**: Updates weights using a single training example at each step. Formula:

$$W = w - \eta \cdot \nabla L(w)$$

- **Advantages**: Simple; requires less memory.

- **Limitations**: Slow convergence; sensitive to learning rate.
- **Use Case**: Suitable for convex optimization problems.

(b) Adam (Adaptive Moment Estimation):

- **Description**: Combines the benefits of momentum and adaptive learning rates.
- **Advantages**: Fast convergence; adaptive to parameter scaling.
- **Limitations**: May not generalize well on some tasks.
- **Use Case**: Widely used in most modern deep learning tasks.

(c) RMSprop:

- **Description**: Maintains a running average of squared gradients to scale the learning rate.
- **Advantages**: Works well with non-stationary objectives.
- **Limitations**: Requires tuning of hyper parameters.
- **Use Case**: Commonly used for recurrent neural networks (RNNs)