**ID: 22308009**

***Name: Arabindo Naha***

**Deep Learningn Assignment**

**Part A: Theoretical Concepts**

1. **Explain Activation Functions**

An **activation function** is a mathematical function applied to the output of a neuron in a neural network. It introduces **non-linearity** into the model, enabling the network to learn and represent complex patterns in the data. Without this non-linearity, a neural network regardless of its depth would behave like a simple linear regression model, limiting its ability to solve complex problems.

The activation function determines whether a neuron should be activated by calculating the weighted sum of its inputs and adding a bias term. This process helps the network make more nuanced decisions and predictions by incorporating non-linear transformations into the neuron's output.

Define and compare the following activation functions:

* **Sigmoid:** Sigmoid Activation function is characterized by ‘S’ shape.This formula ensures a smooth and continuous output that is essential for gradient-based optimization methods.
* It allows neural networks to handle and model complex patterns that linear equations cannot.
* The output ranges between 0 and 1, hence useful for binary classification.
* The function exhibits a steep gradient when x values are between -2 and 2.

**Formula:** σ(x) = 1/ 1+e^(-x)

**Use Cases:** Logistic regression, binary classification.

**Limitations:** Saturation at extreme values leads to vanishing gradients, slower training.

* **ReLU( Rectified Linear Unit):** ReLU activation is defined by A(x) max(0,x), this means that if the input x is positive, ReLU returns x, if the input is negative, it return 0.
* Vale Range:[0,infinit), meaning the function only outputs non-negative value.

**Formula:** f(x) = max(0,x)

**Use Cases:** Most CNN architectures.

**Limitations:** “Dead neurons” for negative inputs.

* **Tanh( Hyperbolic Tangent):** Tanh function or hyperbolic tangent function, is a shifted version of the sigmoid, allowing it to stretch across the y-axis.

**Value Range:** Outputs values from -1 to +1.

**Non-linear:** Enables modeling of complex data patterns.

**Formula:** f(x) = (e^x-e^-x)/(e^x+e^-x)

**Use Cases:** Hidden Layers.

**Use in Hidden Layers:** Commonly used in hidden layers due to its zero-centered output, facilitating easier learning for subsequent layers.

**Limitations:** Similar vanishing gradient issue as Sigmoid.

* **Leaky ReLU:**

**Formula:** f(x) = x if x>0, else f(x)=ax

**Use Cases:** Addresses “dead neurons” in ReLU.

**Limitations:** Introduces slight computational overhead.

1. **Discuss Optimization Algorithms**

Compare the following:

* **SGD( Stochastic Gradient Descent):**

**Advantage:** Simple and easy to implement.

**Limitation:** Can converge slowly or get stuck in local minima.

* **Adam( Adaptive Moment Estimation):**

**Advantage:** Combines momentum and adaptive learning rates, faster convergence.

**Limitation:** Can sometimes generalize poorly.

* **RMSProp (Root Mean Square Propagation):**

**Advantage:** Efficient for non- stationary objectives.

**Limitation:** Can overfit if learning rate aren’t tunes.

b**Learning Rate and its Impact:**

* A **high learning rate** may overshoot the optimal solution, leading to divergence.
* A **Low learning rate** may result in slow convergence or getting stuck in local mimima.

**Modern Techniques to Address Learning Rate Issues:**

* **Learning Rate Schedules:** Gradully reduce the learning rate during training.
* **Adaptive Methods:** Optimizers like Adam adjust learning rate dynamically for each parameter.
* **Warm Restarts:** Periodically reset and reduce learning rate to escape local minima.

**Error Analysis**

Identify and discuss three common errors:

1. **Misclassification in similar classes( e.g.,truck vs. car):**

solution: Use data augumentation to highlight unique features.

1. **Overfitting:**

Solution: Add dropout layers and reduce complexity.

1. **Vanishing Gradients:**

Solution: Use advanced optimizers like Adam and activation function like ReLU.

**Model Design:**

We are required to design a Convolutional Neural Network (CNN) with the following specifications:

* 3- convolutional layers.
* 2 -fully connected layers.
* Incorporate regularization techniques such as dropout and batch normalization to prevent overfiting.

**Conclusion:**

1. Performance After 3 Epochs:

The model achieved a test accuracy of around 48% after training for only 3 epochs.

1. Model Evaluation:

The model’s performance can be further analyzed by reviewing the confusion matrix.

1. Final Thoughts:

This model design provides a solid starting point for a CIFAR-10 image classification task.