Deep Learning Report

Name:	**Ujjwal Kishor Sahoo**
Registration No./Roll No.:	21293
Institute/University Name:	IISER Bhopal
Program/Stream:	Physics
Date:	26 March, 2024

Question 1

Logistic regression seeks an optimal decision boundary in the feature space. The Cross-Entropy loss function aids in this goal by maximizing the likelihood of accurate labels given the model's parameters. This approach considers the probabilistic nature of the problem, unlike Mean Squared Error (MSE), which may yield suboptimal results.

In logistic regression, the output is between 0 and 1, which means it is a probability. Cross Entropy loss quantifies how well the predicted probabilities match the actual labels, ensuring that the model aims to minimize this discrepancy during training.

Cross entropy loss tends to have sharper gradients than mean squared error (MSE) loss, especially when the predicted probabilities are far from the accurate labels. This facilitates faster convergence during training, allowing the model to learn more efficiently.

Regarding the training process, cross-entropy loss typically results in faster convergence and better generalization performance than MSE, especially in scenarios where the classes are imbalanced or the decision boundary is non-linear. At each iteration, the gradients of the loss function with respect to the parameters guide the optimization process towards finding the optimal set of parameters that minimize the discrepancy between the predicted probabilities and the true labels.

Question 2

For a given input X, if all the activation functions in the MLP are linear, then the entire network becomes a linear function of the input X, i.e.

$$\hat{y_i} = w_i X + b_i$$

Now, the loss functions for cross entropy and MSE would be:

$$L_{CE} = -\sum_{i=1}^{2} y_i log(\hat{y_i})$$

$$L_{MSE} = \frac{1}{2}\sum_{i=1}^{2}(y_i - \hat{y_i})^2$$

If we take the value of y_i=1 when it is the index of the correct class, then we have our function defined as the $f(x) = -log(x)$ We need to compute the double derivative of the function to prove whether the function is convex or not:

$$\frac{\partial L}{\partial x} = -\frac{1}{x} \Rightarrow \frac{\partial^2 L}{\partial x^2} = \frac{1}{x^2} > 0$$

For the MSE loss, too we need to calculate the gradient of the function:

$$\nabla^2 \text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \nabla^2 (y_i - \hat{y}_i)^2$$

Since $(y_i - \hat{y}_i)^2$ is a convex function of $\hat{y}_i$, and the sum of convex functions is convex, the MSE loss function is convex.
The answer is option (c) both.

Question 3

1. **Model Architecture**: It takes input size, list of hidden layer sizes, output size, and list of activation functions as arguments. It creates a sequential model with linear layers and specified activation functions. Let X represent the input data, W represent the weight matrices, b represent the bias vectors, and f represent the activation function. The output of each layer can be calculated as follows:

$$Z^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]}$$

$$A^{[l]} = f(Z^{[l]})$$

where $Z^{[l]}$ is the input to layer l, $A^{[l]}$ is the output of layer l, and l denotes the layer index.

2. **Preprocessing**: Preprocessing input images may involve normalization of pixel values to a specific range (e.g., [0, 1]) and resizing images to a fixed size, which means normalizing them.

3. **Data Loaders**: We create data loaders for training and testing datasets using `keras.datasets.cifar10.loaddata`.

4. **Training Loop**: We train the model using a simple training loop. We iterate through batches of data, perform forward passes, calculate loss and backward passes, and update weights using the Adam optimizer.

5. **Evaluation**: We evaluate the trained model on the test dataset and print the accuracy.

Hyperparameter tuning strategies include grid search, random search, and more advanced techniques like Bayesian optimization. These methods aim to find the optimal set of hyperparameters that maximize the neural network's performance. Techniques like learning rate schedules, weight decay, dropout, and Batch normalization can be employed to improve performance and generalize the model.

Question 4

LeNet-5: It can be used for the SVHN dataset due to its simplicity and efficiency. However, it may not perform well compared to more complex models like VGG or ResNet. It achieves a test accuracy of 46
AlexNet: It's deeper architecture requires more computational resources than simpler models like LeNet-5. Its test and train scores are better than LeNet5; this is because it is a more complex model.
VGG: VGG can be well suited for the SVHN dataset due to its strong performance on image classification tasks and its ability to capture intricate features from the data. However, its deeper architecture may require more computational resources than shallower models like LeNet-5. The accuracy is 81
ResNet: ResNet can be highly suitable for the SVHN dataset due to its state-of-the-art image classification performance and ability to capture intricate features effectively. However, its more profound architecture may require significant computational resources for training.