Deep Learning Assignment 1

Name:	**Chavan Vishwaraj Sopan**
Registration No./Roll No.:	21086
Institute/University Name:	IISER Bhopal
Program/Stream:	DSE
Date of Submission:	March 27, 2024

Question 1

In logistic regression, the cross-entropy loss function is typically preferred over mean squared error (MSE) for several reasons, including its ability to provide a clearer and more optimal solution.

Nature of Logistic Regression:
Logistic regression is commonly used for binary classification problems, where the target variable has two possible outcomes (e.g., 0 or 1). The logistic function (also known as the sigmoid function) is used to model the probability that a given input belongs to one of the classes.

Interpretability:
Cross-entropy loss directly relates to the likelihood of the predicted probabilities matching the actual labels. It's intuitive to interpret: the lower the cross-entropy loss, the better the model's predictions align with the true labels. In contrast, MSE doesn't directly relate to the probabilities. It's more suited for regression tasks where the output is continuous and measuring the squared difference between predicted and actual values is meaningful.
Gradient Descent Optimization:

Cross-entropy loss leads to smoother gradients during optimization compared to MSE, especially when used in conjunction with the logistic activation function. In logistic regression, optimizing cross-entropy loss through techniques like gradient descent tends to converge faster and more reliably to the global optimum due to its convex nature.

Single Optimal Solution:
Cross-entropy loss guarantees a single best solution due to its convexity. There's no ambiguity in the optimization process, and the model converges to a unique set of parameters that minimize the loss function. With MSE, especially in logistic regression where it's not the ideal loss function, there might be multiple local minima, leading to potential confusion and suboptimal solutions.

Question 2

In a binary classification task with a deep neural network equipped with linear activation functions, the loss function that guarantees a convex optimization problem is (b) Mean Squared Error (MSE).

Mean Square Error

In binary classification, let y denote the target label (either 0 or 1) and $\hat{y}$ denote the predicted output of the model. With linear activation functions, the output of the model is a linear combination of inputs and weights, i.e

$$\hat{y} = WX + b$$

As the output of the model is a linear function of the inputs, the MSE loss function becomes a quadratic function of the parameters (weights and biases). Quadratic functions are convex, meaning they have a single global minimum. Thus, using MSE as the loss function guarantees convexity in the optimization problem.

Cross Entropy
While Cross Entropy loss is commonly used for binary classification, it's not guaranteed to result in a convex optimization problem when combined with linear activation functions. The nonlinearity introduced by the logarithm function makes the loss function non-convex, and the optimization problem may have multiple local minima.

Question3

We define a simple feedforward neural network (FNN) with three dense layers.
The input images are preprocessed using torchvision transforms, which convert images to tensors and normalize them.
We use the MNIST dataset for training and testing.
The FNN model is trained using stochastic gradient descent (SGD) with cross-entropy loss.
We evaluate the trained model on the test set and calculate its accuracy.
Hyperparameters such as learning rate, number of hidden layers, number of neurons per layer, and activation functions (ReLU in this case) can be tuned to optimize the performance of the model. Techniques like grid search or random search can be employed for hyperparameter tuning. Additionally, techniques like dropout or batch normalization can be used to prevent overfitting and improve generalization.

Question4

The provided code implements a classifier for the Street View House Numbers (SVHN) dataset using various pre-trained models such as LeNet-5, AlexNet, VGG, ResNet (18, 50, 101). The evaluation includes training each model on the SVHN dataset and testing its performance.

1 Code Structure

1. **Data Loading and Transformation:** The SVHN dataset is loaded using torchvision's `datasets.SVHN` module. Data transformation includes resizing to 32x32, converting to tensor, and normalizing.

2. **Model Architectures:** The code defines custom implementations of LeNet-5 and AlexNet architectures along with loading pre-trained VGG and ResNet models.

3. **Loading Pretrained Weights:** Pretrained weights are loaded for AlexNet, VGG, and ResNet models, and custom output

layers are added to match the number of classes in the SVHN dataset.

4. **Training and Testing:** The code defines functions for training (`train_model`) and testing (`test_model`) the models. Training is performed using cross-entropy loss and stochastic gradient descent optimizer.

5. **Evaluation:** Each model is trained for 5 epochs, and its performance is evaluated on the test set. The accuracy is reported for each model.

2 Results

- **LeNet-5:** Achieved a test accuracy of 47.68
- **AlexNet:** Achieved a test accuracy of 94.52
- **VGG:** Achieved a test accuracy of 94.52
- **ResNet-18:** Achieved a test accuracy of 92.24
- **ResNet-50:** Achieved a test accuracy of 93.11
- **ResNet-101:** Achieved a test accuracy of 92.47
- **Performance Comparison:** Among the models evaluated, AlexNet and VGG achieved the highest test accuracies, followed by ResNet-50. LeNet-5 performed the poorest, indicating that deeper and more complex architectures generally perform better on the SVHN dataset.
- **Effectiveness of Pretraining:** Pretrained models (AlexNet, VGG, ResNet) generally outperformed the custom LeNet-5 architecture, highlighting the effectiveness of pretraining on large-scale datasets like ImageNet.
- **ResNet Variants:** ResNet-50 outperformed ResNet-18 and ResNet-101, suggesting that ResNet-50 strikes a good balance between model complexity and performance for the SVHN dataset.

3 Conclusion

The evaluation demonstrates the importance of selecting appropriate model architectures and leveraging pretraining for achieving higher accuracy on the SVHN dataset. Among the models evaluated, AlexNet and VGG show the best performance, indicating their suitability for image classification tasks like SVHN.