Q1. Which loss function, out of Cross Entropy and Mean Squared Error, works best with logistic regression because it guarantees a single best answer (no room for confusion)? Explain why this is important and maybe even show how it affects the model's training process. [$3Marks$] [$Theory$]

Ans. The cross-entropy is goord over Mean Squarred Error (MSE) for logistic regression. In Logistic regression, the output is lie between 0 to 1, repesnting the likelihood of a particular instance to a certain class.This output is obtained by passing the weighted sum of the features through a sigmoid function. Cross Entropy measures the difference between the predicted probabilites and the actual labels.It calculates the loss as the negative log-likelihood of the obeserved data given the model's parameter.

$$K(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \tag{1}$$

where y is the true label belong to 0 to 1, $\hat{y}$ is the predicted probability and N is the number of samples. Cross Entropy loss ensures that the model aims to predict the probabilities as close to the true labels as possible. It penalizes the model more heavily for confidently incorrect predictions, thus pushing the model to learn better representations. while, In MSE loss measures the squared difference between the predicted values and the true labels. While it's a suitable loss function for regression tasks where the output is continuous, it's not ideal for logistic regression because it does not directly optimize for probabilities.

Q2. For a binary classification task with a deep neural network (containing at least one hidden layer) equipped with linear activation functions, which of the following loss functions guarantees a convex optimization problem? Justify your answer with a formal proof or a clear argument. (a) CE (b) MSE (c) Both (A) and (B) (d) None [$3Marks$] [$Theory$]

Ans.When using a deep neural network (DNN) for binary classification with linear activation functions, the Mean Squared Error (MSE) loss function is the best choice. This is because MSE ensures convex optimization, which means the optimization process is more efficient and reliable.

The linearity of the activation functions in the hidden layers of the DNN makes the optimization landscape convex, allowing for easier and more effective parameter optimization. Additionally, the nature of the MSE loss function, with its squared term, reinforces this convexity by penalizing deviations from the true labels in a convex manner. This means that any local minimum encountered during optimization is also the global minimum, ensuring that the optimization process converges to the best solution.

On the other hand, the Cross-Entropy (CE) loss function, typically used in classification tasks, does not guarantee convexity, especially when linear activation functions are used. While CE loss functions work well with nonlinear activation functions like sigmoid or softmax, their convexity depends on these choices. This means that in scenarios where linear activation functions are used, the CE loss function may not ensure convex optimization.

In summary, for binary classification tasks using linear activation functions, the MSE loss function is preferred because it guarantees convex optimization, ensuring efficient training and convergence to the best solution. So answer is (b)

Q3. Dense Neural Network: Implement a feedforward neural network with dense layers only. Specify the number of hidden layers, neurons per layer, and activation functions. How will you preprocess the input images? Consider hyperparameter tuning strategies. [2 for implementation and 3 for explanation] [Code and Report]

Ans.Code

Report

Introduction

For the CIFAR-10 dataset, we implemented a feedforward neural network with dense layers only. We specified the number of hidden layers, neurons per layer, and activation functions. We also considered preprocessing strategies for the input images and hyperparameter tuning strategies.

Model Architecture

We used 3 hidden layers in which each hidden layer have 128 neurons, Relu activation function was used for all hidden layers and softmax activation function was used for the output layer.

Hyperparameter Tuning

We performed hyperparameter tuning using GridSearchCV to find the best combination of activation function and neurons per layer. We explored different values for these hyperparameters and selected the ones that yielded the best performance.

Results

Best model is ReLU, Neuran per layer=128 and the best accuracy is 35.24%

Conclusion

The feedforward neural network with dense layers achieved a best accuracy of 35.24% on the CIFAR-10 dataset. This model provides a baseline for further experimentation with different architectures, preprocessing techniques, and hyperparameter tuning strategies to improve performance.

Q4. Build a classifier for Street View House Numbers (SVHN) (Dataset) using pretrained model weights from PyTorch. Try multiple models like LeNet-5, AlexNet, VGG, or ResNet (18, 50, 101). Compare performance comment why a particular model is well suited for SVHN dataset. (You can use a subset of dataset (25%) in case you do not have enough compute.) [4 Marks] [Code and Report]

Ans. code

Report

Introduction

The report summarize the various deep learning models on the SVHN dataset using pretrained architectures.

Dataset

The Street View House Numbers (SVHN) dataset consists of images of house numbers collected from Google Street View. It contains over 600,000 labeled digits.

Models

LeNet-5, VGG-16, ResNet(18,50,101)

Training and Evaluation

Each model was trained for 10 epochs using the Adam optimizer with a learning rate of 0.001. Training included data augmentation techniques such as random rotation, random cropping, and horizontal flipping.

Results

Conclusion

- The LeNet-5 and ResNet-18 models performed well on the SVHN dataset, achieving test accuracies of 75.56% and 88.23% respectively.

Model	Test Accuracy	Precision	Recall	F1-score
LeNet-5	75.56%	0.7361	0.7324	0.7300
VGG-16	18.43%	0.0995	0.1000	0.0328
ResNet-18	88.23%	0.8746	0.8727	0.8729
ResNet-50	88.50%	0.8796	0.8726	0.8745
ResNet-101	83.98%	0.8310	0.8339	0.8316

- VGG-16 showed poor performance on this task, likely due to its large number of parameters relative to the size of the dataset.
- ResNet-50 and ResNet-101 achieved moderate performance, with accuracies of 88.50% and 83.98% respectively.

When comparing the performance of the models on the SVHN dataset, we can see that LeNet-5 and ResNet-18 perform better than VGG-16, ResNet-50, and ResNet-101 in terms of test accuracy, precision, recall, and F1-score.

In conclusion, LeNet-5's architecture, parameter efficiency, and the use of data augmentation techniques make it well-suited for the SVHN dataset. Despite being a relatively simple convolutional neural network (CNN) architecture, LeNet-5 achieves a test accuracy of 75.56% on the SVHN dataset, outperforming deeper and more complex architectures like VGG-16, ResNet-50, and ResNet-101. This success can be attributed to LeNet-5's design for handwritten digit recognition, which aligns well with the nature of the SVHN dataset. Additionally, the smaller number of parameters in LeNet-5 compared to the more modern architectures may have helped prevent overfitting on the limited training data available in the SVHN dataset. The use of data augmentation techniques further enhances LeNet-5's ability to generalize well to unseen data.