MACHINE LEARNING

Assignment #1

Chinmay Jain
SRNo: 23006
MTech(Coursework) CSA, IISc
April,7,2024

1 Softmax regression:

1.1 Mathematical Background

Softmax function is applied on the output layer of logistic regression to convert raw predictions into probabilities. Given an input vector z (the output of the logistic regression model), the softmax function computes the probability distribution over multiple classes. For a single example, the softmax function is defined as:

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

1.2 Loss Function

The Loss function used binary classification is binary cross-entropy loss, given by:

$$-\frac{1}{N}\sum_{i=1}^{N}[y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)})]$$

For Multi class classification, the categorical cross-entropy loss is used as:

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_k^{(i)}\log(\hat{y}_k^{(i)})$$

1.3 Training Strategy

1. **Batch Size Selection:** The batch size chosen was 1024 .Larger batch size offers computational efficiency at the cost of reduced stochasticity, while smaller batch sizes introduce noise but offer better generalization.

2. **Image Sampling:** The training and validation data are shuffled and then split into training and validation sets of ratio 80:20. This ensures randomness in the batches presented to the model during training, reducing the risk of model overfitting .

3. **Training Stopping Criteria:** I have not used any explicit stopping criterion. But, training could be stopped based on some convergence criteria like change in loss below a threshold or after a fixed number of iterations .

4. **Learning Rate Schedule:** The learning rate (`lr`) is decayed by a factor of 0.2 every 100 iterations with the initial learning rate of 1e-3. This technique, known as learning rate annealing, helps in fine-tuning the model and avoiding overshooting.

5. **Gradient Clipping:** Gradient clipping (`grad_norm_clip`) is applied to prevent exploding gradients, a common issue especially in deep learning models. If the norm of gradients exceeds a threshold, they are rescaled to ensure stability during training.

6. **Regularization:** L_2 regularization (`l2_lambda`) is applied to the weights during gradient calculation to prevent overfitting by penalizing large weight values L2 lambda value was taken as 0.1.

1.4 Results

The best validation accuracy achieved using softmax was 36.89 percent.

The graphs of accuracy and losses are shown in Fig1 an Fig2.

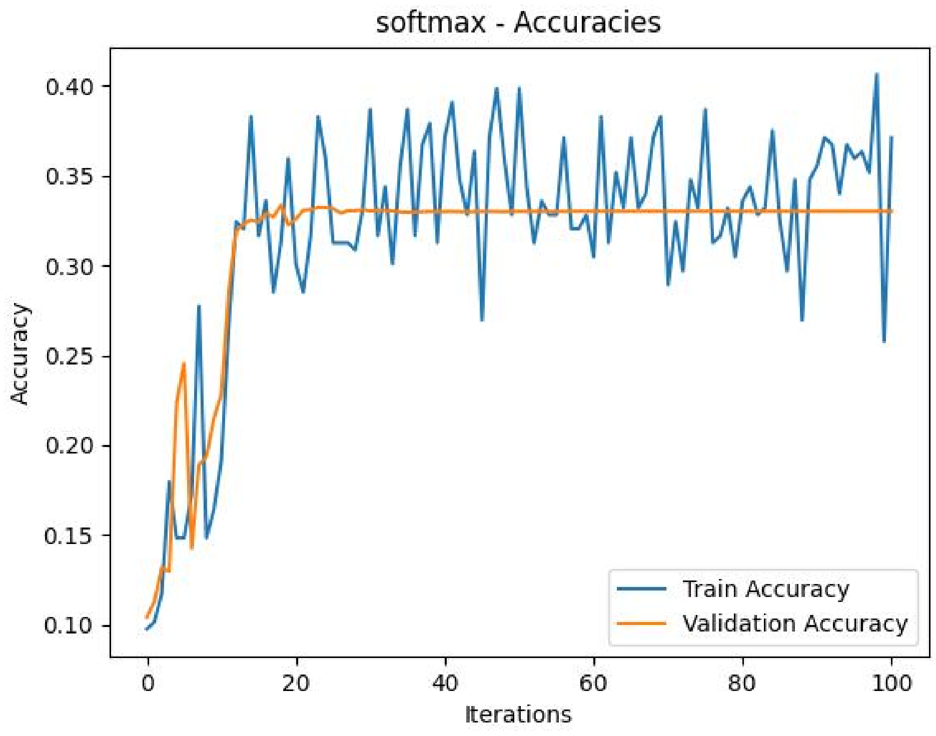


Figure 1: Accuracies vs Iterations for softmax regression

]

2 Contrastive Representation learning

2.1 Encoder Architecture:ALexNet

- **Type**: Convolutional Neural Network (CNN)
- **Convolutional Layers**:
  - Input channels: 3 (for RGB images)
  - Output channels: 64
  - Kernel size: 3×3
  - Stride: 2
  - Padding: 1
  - Activation function: ReLU
  - Batch normalization is applied after ReLU activation.

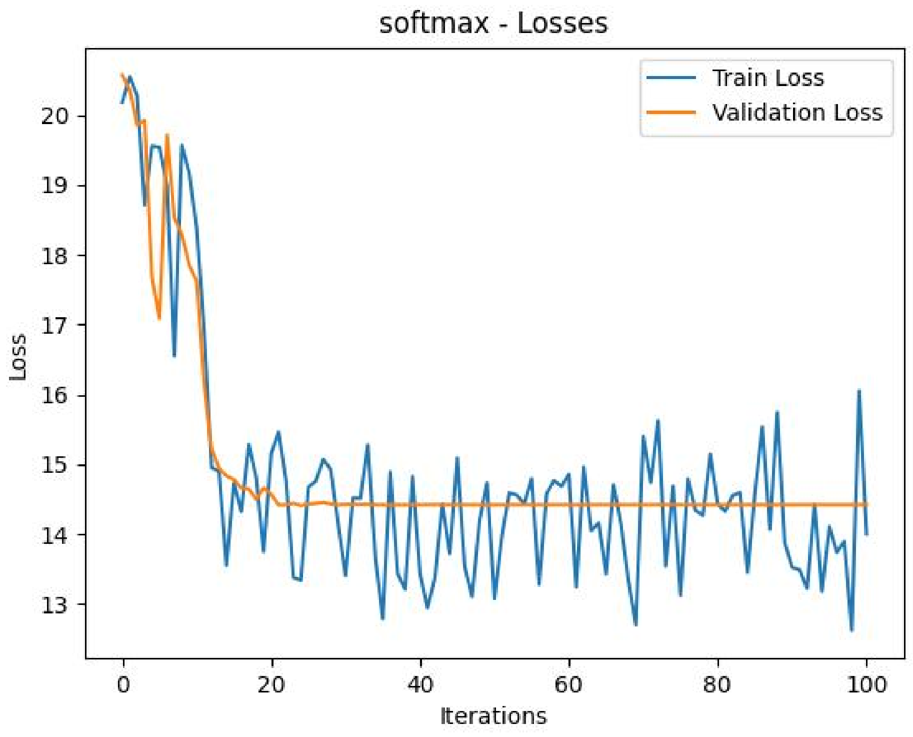


Figure 2: Loss vs Iterations for Softmax regression

- **Activation Function**: ReLU (Rectified Linear Unit) is used after each convolutional layer in the feature extraction part and the fully connected layers in the classifier part, except for the output layer where softmax is used.
- **Dropout**: Dropout is applied before the fully connected layers in the classifier part.
- **Batch Normalization**: Batch normalization is applied after each convolutional layer in the feature extraction part and before the fully connected layers in the classifier part.
- **dout** : Dimensional output vectors selected as 1000.

2.2 Classifier Model Description

The classifier model is a simple feedforward neural network that takes as input the latent features (encoded representation) produced by the encoder. Here's a breakdown of the classifier architecture:

- **Fully Connected Layers**:
  - Input dimension: z_{dim} (size of the latent space)
  - Output dimension: 32
  - Activation function: ReLU

- **Output**:
    - The output layer applies softmax activation to produce class probabilities.

The training strategy was same as softmax implementation.

2.3 Loss Function:Triplet Loss

Triplet loss is the loss function used for contrastive learning for training neural networks to learn embeddings . The main idea behind triplet loss is to pull together embeddings of similar examples while pushing apart embeddings of dissimilar examples.The loss function encourages the network to minimize the distance between the anchor and the positive example while simultaneously maximizing the distance between the anchor and the negative example by a margin.

Mathematically, the triplet loss can be defined as follows:

$$L_{\text{triplet}} = \max\{\|f(a) - f(p)\|_2^2 - \|f(a) - f(n)\|_2^2 + \alpha, 0\}$$

Where:

- $f(\cdot)$ represents the embedding function (the output of the neural network).
- a represents the anchor example.
- p represents the positive example (similar to the anchor).
- n represents the negative example (dissimilar to the anchor).
- $\|\cdot\|_2$ denotes the Euclidean distance.
- α is a margin hyperparameter that controls the minimum difference between the distances of the anchor-positive pair and the anchor-negative pair.

2.4 Training strategy

The training strategy for the provided encoder and classifier models on the CIFAR-10 dataset involves several key steps. Firstly, the CIFAR-10 dataset, which consists of 60,000 32x32 color images in 10 classes, is split into training and validation sets. The encoder model is trained on the training set, where it learns to extract high-level features from the input images through a series of convolutional layers followed by max-pooling operations and batch normalization. The classifier model then takes these extracted features and learns to classify them into the respective classes using fully connected layers with ReLU activation and softmax output. During training, dropout regularization is applied to prevent overfitting, and batch normalization helps stabilize and speed up the training process. The training process involves optimizing the parameters of both the encoder and classifier models using a suitable optimization algorithm such as stochastic gradient descent (SGD) or Adam. The training is typically performed for multiple epochs, with the model's performance monitored on the validation set to prevent overfitting and ensure generalization to unseen data. Finally, once the training process is complete, the performance of the trained model can be evaluated using various metrics such as accuracy, precision, recall, and F1-score on a separate test set.

2.5 Hyperparameters

For contrastive representation learning:

- Number of iterations for training: 5000
- Batch size for training :1024
- Learning rate for optimization: 0.003
- Number of iterations for training=5000
- Batch size for training :1024
- Learning rate for optimization:0.001
- L2 regularization lambda parameter: 0.1(default)
- Gradient norm clipping threshold:4.0 (default)

2.6 Performance

	Fine Tune Linear	Fine Tune NN
Train Set	0.2685	0.9785
Val Set	0.2465	0.8436

Table 1: Accuracy Table after 2000 iterations

As evident from table and the graphs also that NN classifier works better than Linear classifier fine tuning models.This is because nn has more capacity to learn complex patters compared to logistic regression leading to better performance

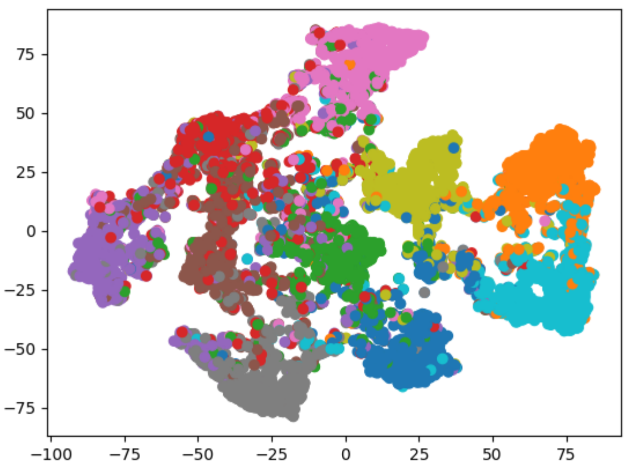
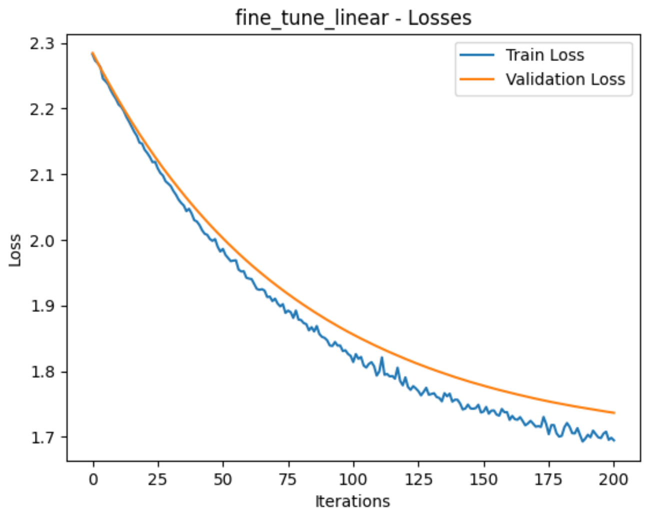
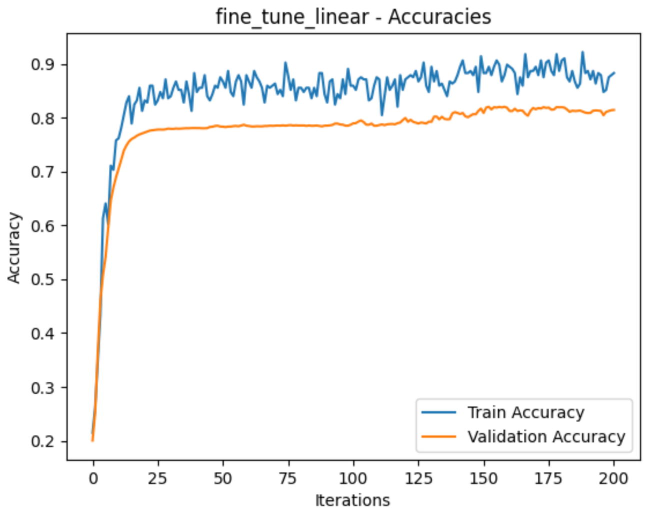


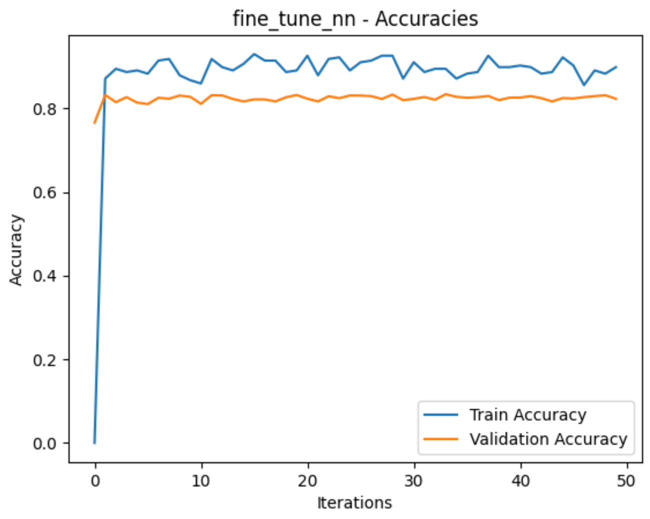
Figure 3: t-SNE plot

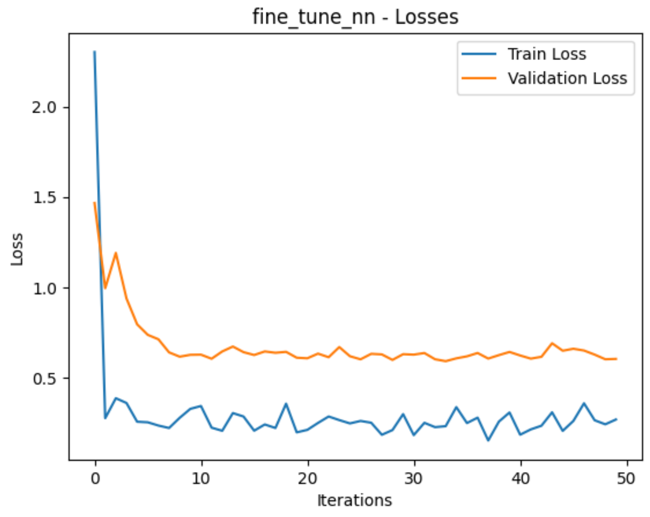
In the 2-D t-SNE(t-distributed Stochastic Neighbor Embedding) graph we can observe some clusters of different colours forming of similar data points.The clusters formed are of data pointsof different data points.It helps us to identify patterns between different groups.Dense regions correspond to areas where many similar data points are located, and the less dense regions indicate outliers areas of the data.fine-tuning allows the network to adapt its parameters, which can further enhance performance compared to using fixed features with logistic regression. Better the separation between the clusters better is the model's performance.

The best accuracy obtained on leaderboard by NN model is 82.38.






fine_tune_nn - Accuracies
Accuracy
Iterations
Train Accuracy
Validation Accuracy
0.0
0.2
0.4
0.6
0.8
0
10
20
30
40
50


fine_tune_nn - Losses
Loss
Iterations
Train Loss
Validation Loss
0.5
1.0
1.5
2.0
0
10
20
30
40
50