Machine Learning Assignment 1

Alokam Gnaneswara Sai
22582

April 7, 2024

Question 1)1

Have uploaded the file .

Question 1)2

Cost Function

The cost function is defined as the cross-entropy loss function, which is commonly used for classification tasks. It is given by:

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i + \epsilon) + (1 - y_i)\log(1 - \hat{y}_i + \epsilon)\right] \qquad \text{(Binary Cross-Entropy Loss)}$$

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij} + \epsilon)\right] \qquad \text{(Categorical Cross-Entropy Loss)}$$

N : Number of samples
C : Number of classes
y_i : True label of the ith sample
$\hat{y}_i$: Predicted probability of the positive class for the ith sample in binary classification
y_{ij} : True label of the ith sample for the jth class in multi-class classification
$\hat{y}_{ij}$: Predicted probability of the jth class for the ith sample in multi-class classification
ϵ : Small constant (e.g., 1×10^{-10}) added to prevent taking the logarithm of zero

The goal of the training process is to minimize this cost function by finding the optimal values for the parameters θ.

Training Strategy

For the training strategy, a batch size of 256 has been chosen.The chosen batch size of 256 was determined through experimentation. A low batch size could lead to noisy updates and slower convergence . During each training iteration, the model updates based on the average gradient computed from 256 randomly sampled training examples. .

The model updates are performed using Stochastic Gradient Descent (SGD) with a learning rate of 1e-3. Additionally, a l2 regularization lambda value of 0.1 is applied to prevent overfitting by penalizing large parameter values. The training process is carried out for 2000 iterations. Gradient clipping is applied to prevent exploding gradients, ensuring stable training.

Image Sampling

During each iteration, a batch of images is sampled randomly from the training dataset. With a batch size of batch-size, a new batch of batch-size images is selected in each iteration. Since the training process runs for 1000 iterations, and the CIFAR-10 dataset consists of 50,000 training images, in expectation, the model will traverse over the entire dataset approximately 5 times during the training process. This random sampling strategy ensures that the model encounters a diverse set of images during training, promoting better generalization.

Training stopping criterion

The criteria for stopping the training involved monitoring the validation loss. If the validation loss did not decrease or began to increase consistently over several iterations, training was stopped early to prevent overfitting. Additionally, an early stopping mechanism was implemented, where training was halted if the validation loss continued to increase for a predefined number(5-6) of consecutive iterations.

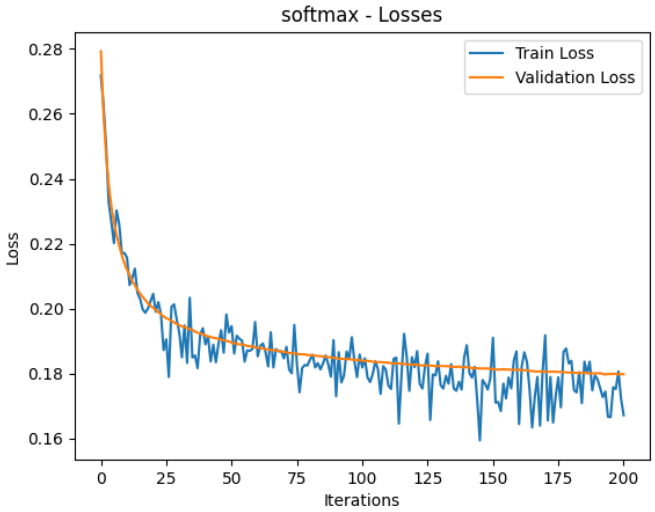

Figure 1: Variation of loss with iterations during model training

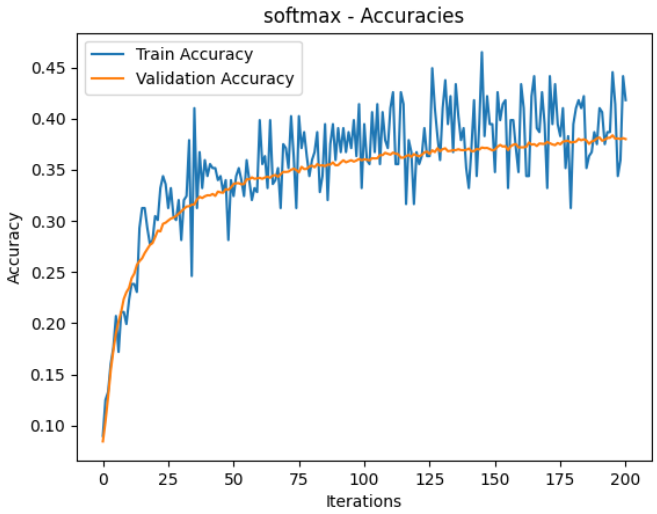

Figure 2: Variation of accuracies with iterations during model training

Question 2)1

Architecture

- **Dimension** $d_{out} = 64$
- **Activation Function:** ReLU (Rectified Linear Unit)
- **Batch size** : 1024

Choosing an encoded dimension of 64 strikes a balance between capturing sufficient information from the $32 \times 32 \times 3$ input images and avoiding redundancy in the encoded vectors. Smaller vector dimensions may not capture the details of the image .
Conversely, larger vector dimensions may introduce unnecessary redundancy, potentially leading to increased computational overhead and slower convergence during training. By selecting a dimension of 64, the encoded vectors effectively capture relevant features of the input images.

Layer	Type	Output Size
1	Conv2d (3, 64, kernel_size=3, padding="same")	32x32x64
	BatchNorm2d (64)	32x32x64
	ReLU (inplace=True)	32x32x64
2	Conv2d (64, 64, kernel_size=3, padding="same")	32x32x64
	BatchNorm2d (64)	32x32x64
	ReLU (inplace=True)	32x32x64
	MaxPool2d (kernel_size=2, stride=2)	16x16x64
3	Conv2d (64, 128, kernel_size=3, padding="same")	16x16x128
	BatchNorm2d (128)	16x16x128
	ReLU (inplace=True)	16x16x128
4	Conv2d (128, 128, kernel_size=3, padding="same")	16x16x128
	BatchNorm2d (128)	16x16x128
	ReLU (inplace=True)	16x16x128
	MaxPool2d (kernel_size=2, stride=2)	8x8x128
5	Conv2d (128, 256, kernel_size=3, padding="same")	8x8x256
	BatchNorm2d (256)	8x8x256
	ReLU (inplace=True)	8x8x256
6	Conv2d (256, 256, kernel_size=3, padding="same")	8x8x256
	BatchNorm2d (256)	8x8x256
	ReLU (inplace=True)	8x8x256
7	Conv2d (256, 256, kernel_size=3, padding="same")	8x8x256
	BatchNorm2d (256)	8x8x256
	ReLU (inplace=True)	8x8x256
	MaxPool2d (kernel_size=2, stride=2)	4x4x256
8	Conv2d (256, 512, kernel_size=3, padding="same")	4x4x512
	BatchNorm2d (512)	4x4x512
	ReLU (inplace=True)	4x4x512
9	Conv2d (512, 512, kernel_size=3, padding="same")	4x4x512
	BatchNorm2d (512)	4x4x512
	ReLU (inplace=True)	4x4x512
10	Conv2d (512, 512, kernel_size=3, padding="same")	4x4x512
	BatchNorm2d (512)	4x4x512
	ReLU (inplace=True)	4x4x512
	MaxPool2d (kernel_size=2, stride=2)	2x2x512
11	Conv2d (512, 512, kernel_size=3, padding="same")	2x2x512
	BatchNorm2d (512)	2x2x512
	ReLU (inplace=True)	2x2x512
12	Conv2d (512, 512, kernel_size=3, padding="same")	2x2x512
	BatchNorm2d (512)	2x2x512
	ReLU (inplace=True)	2x2x512
13	Conv2d (512, 512, kernel_size=3, padding="same")	2x2x512
	BatchNorm2d (512)	2x2x512
	ReLU (inplace=True)	2x2x512
14	Conv2d (512, 512, kernel_size=3, padding="same")	2x2x512
	BatchNorm2d (512)	2x2x512
	ReLU (inplace=True)	2x2x512
	MaxPool2d (kernel_size=2, stride=2)	1x1x512
15 (FC)	Linear (512, 256)	256
	ReLU (inplace=True)	256
	Dropout (0.5)	256
	Linear (256,d_out)	d_out

Layer	Type	Output Size
0	Input (d_out)	d_out
1	Linear (d_out 10)	10

Table 2: Classifier Architecture

Question-2)2

The loss function employed is Triplet Margin Loss.It is defined as follows

$$L(a, p, n) = \max\{\|\mathbf{a}_i - \mathbf{p}_i\|_p - \|\mathbf{a}_i - \mathbf{n}_i\|_p + \text{margin}, 0\}$$

where

- $\mathbf{a}_i$ represents the anchor sample,
- $\mathbf{p}_i$ represents the positive sample,
- $\mathbf{n}_i$ represents the negative sample, and
- $\|\mathbf{x}_i - \mathbf{y}_i\|_p$ represents the p-norm distance between $\mathbf{x}_i$ and $\mathbf{y}_i$.

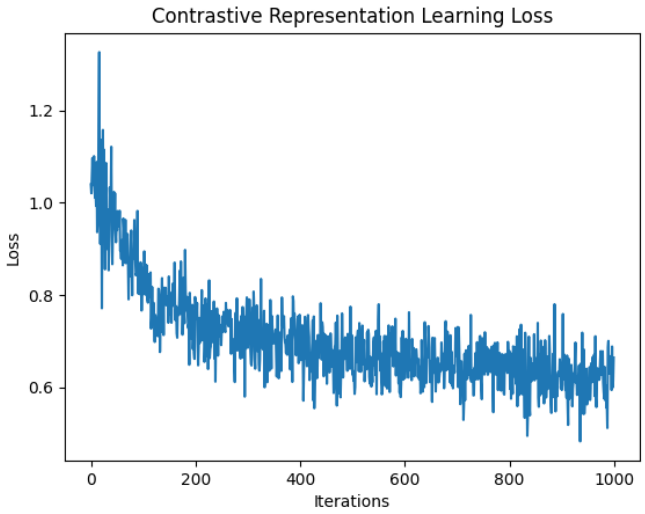

Figure 3: Variation of losses with iterations during model training

The model ran for 2000 iterations where in each iteration, batches of batch_size images were yielded. These batches consisted of triplets of images (I_a, I_b, I_c), where I_a and I_b were randomly selected images with the same label (Anchor and Positive samples), and I_c was a randomly selected image with a different label (Negative sample). This sampling process ensured that the model learned to maximize similarity between v_a and v_b while minimizing similarity between v_a and v_c, as per the training objective.

I chose a batch size of 512 because the training dataset of CIFAR-10 consists of 40,000 images. To run for 20 epochs, covering the entire dataset once in each epoch, a larger batch size is beneficial. With a batch size of 512, each epoch requires approximately 79 iterations to process all images in the dataset once ($\frac{40{,}000}{512} \approx 78.125$). Therefore, for the total of 20 epochs, the model would run for approximately 1580 iterations (20 epochs $\times$ 79 iterations/epoch $\approx$ 1580 iterations). By running for 2000 iterations, the model will converge towards an optimal solution. Additionally, using a larger batch size often results in better utilization of hardware resources, such as GPU memory and processing power, thereby improving training efficiency.

Question-2)3

Observations from the t-SNE plot indicate that all 10 classes in CIFAR-10 are distinctly separated. This separation indicates that the learned embedded vectors of images belonging to the same class are clustered closely together where as vectors belongs to different classes are seperated which indicates that the encoder has learnt the representations correctly.

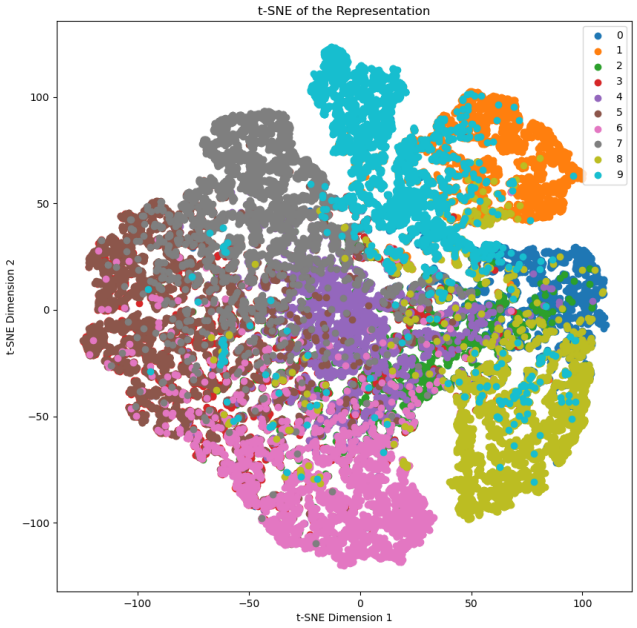

Figure 4: TSNE plot

Question-2)4

Hyperparameters:

- Training Accuracy 70%
- Validation accuracy: 60%
- Batch size: 1024
- Number of iterations: 3000
- Learning rate: 1×10^{-3}
- L_2 lambda : 0.1

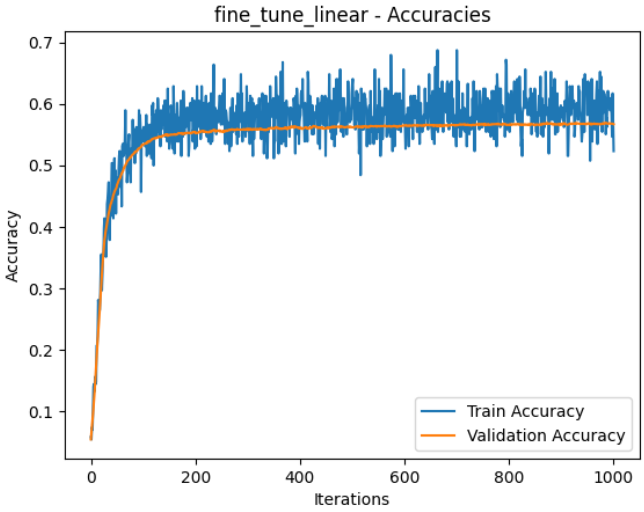

Figure 5: Variation of accuracies with iterations during model training

Question-2)5

The classification accuracy of Network B on the training set is 98%, while on the validation set, it is 85%. The hyperparameters used for training are as follows:

- Batch size: 512
- Learning rate: 0.0009

Fine-tuning a softmax classifier using learned embedding vectors may not be as effective as training a neural network classifier that maps embedding vectors directly to classes because Softmax regression models have limited flexibility compared to neural networks. They are linear models and can only learn linear decision boundaries, which may not capture complex patterns in the data.

When fine-tuning a softmax classifier using learned embedding vectors, the model is essentially constrained by the linear nature of softmax regression. It may struggle to capture the relationships between embedding vectors and class labels, especially in datasets with complex and non-linear class distributions.

On the other hand, a neural network classifier trained directly on embedding vectors can leverage its non-linear layers to learn more expressive representations and better capture the underlying structure of the data. This allows it to achieve higher performance on classification tasks, especially in scenarios where the class boundaries are non-linear and complex.

In Question -1 where we do not learn representations and simply flatten the image, the performance can is even worse. Flattening the image discards the spatial information present in the image and treats each pixel as an independent feature. This approach fails to capture the spatial relationships and hierarchical structures present in the data.

On the other hand, using an encoder architecture to obtain a vector representation of the image preserves spatial information.

Therefore, compared to simply flattening the image, using an encoder architecture to learn representations provides a more effective way to encode the spatial information present in images, leading to better performance.

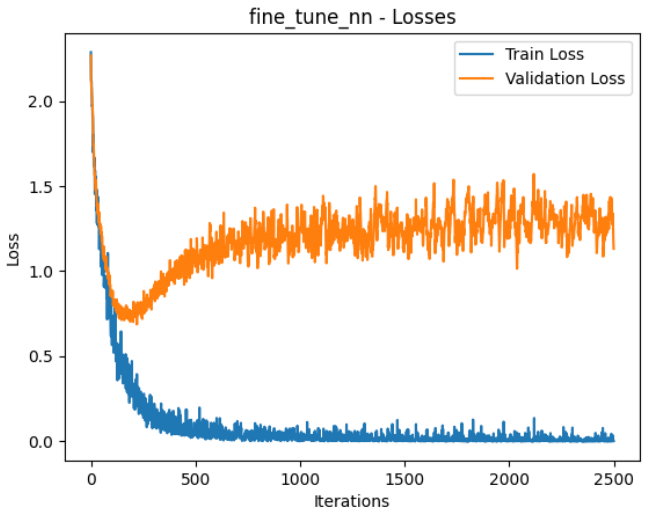

Figure 6: Variation of losses with iterations during model training for fine-tune-nn

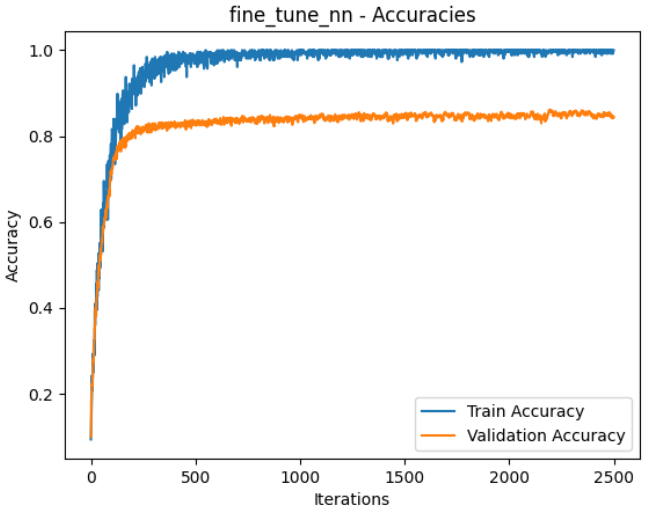

Figure 7: Variation of accuracies with iterations during model training

Note

The best accuracy I got on leader-board is 84.98%