ML Assignment 1

Yaswanth Kumar Pattapu

22523

Problem 1: Binary Classification

All the TODO sections have been completed.

Parameter	Value
Learning Rate	0.001
Batch Size	256
L2 Regularization Lambda	0.1
Number of Iterations	1000

Table 1: Parameter Values for Binary Classification

The Binary Cross-Entropy Loss is used as a cost function in Binary Classification. The model attained an accuracy of **80.55%** with the specified parameter values for Binary Classification.

Multi-Class Logistic Regression(Softmax Regression)

Cost Function

We use Cross entropy loss as a Cost function. The softmax cost function is similar to logistic regression, except that we now sum over the K different possible values of the class label. Cross entropy is a measure of how well a set of estimated class probabilities match the target classes. So, we can say that it measures the performance of the model.

$$cross\_entropy\_loss = -(1/m) * \sum_{i=1}^{m} \sum_{k=1}^{K} y_k^i log(y\_pred_k^i) \tag{1}$$

y_k^i is the target probability that says whether the instance i belongs to class k or not. The probability will be equal to 1 or 0 depending on which class the instance belongs.

K is the number of classes

Training Strategy

Batch Size

When setting up the batch size we need to consider the size of the dataset, available memory and computational resources. The batch size refers to the number of training examples utilized in one iteration. A smaller batch size can lead to a noisy gradient signal, but it allows for more frequent weight updates. On the other hand, a larger batch size provides a more accurate estimate of the gradient, but with less frequent weight updates.

Started with a small batch size that can fit in our system and can provide a reasonable balance between speed and accuracy. A small batch size ensures that each training iteration is very fast, and although a large batch size will give a more precise estimate of the gradients. The batch size of **256** was chosen after conducting multiple experiments

Image Sampling

Random sampling means selecting images randomly from the dataset for each training round. This helps the model see different types of images in each batch, which stops it from focusing too much on specific examples, which helps prevent overfitting.

Stopping Criteria for Training

When training a large network, there will be a point during training when the model will stop generalizing and start learning the statistical noise in the training dataset. This overfitting of the training dataset will increase generalization error, making the model less useful for making predictions on new data.

During training, the model is evaluated on a holdout validation dataset after each epoch. If the performance of the model on the validation dataset starts to degrade (e.g. loss begins to increase or accuracy begins to decrease), then the training process is stopped.

The trigger will use a monitored performance metric to decide when to stop training. This is often the performance of the model on the holdout dataset, such as the loss.

In the simplest case, training is stopped as soon as the performance on the validation dataset decreases as compared to the performance on the validation dataset at the prior training epoch (e.g. an increase in loss).

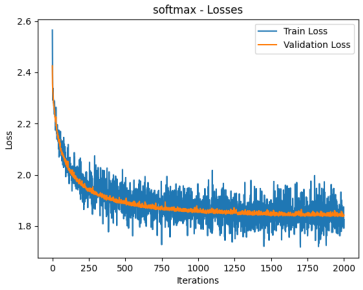
Some more elaborate triggers may include:

- No change in metric over a given number of epochs.
- An absolute change in a metric.
- A decrease in performance observed over a given number of epochs.
- Average change in metric over a given number of epochs.

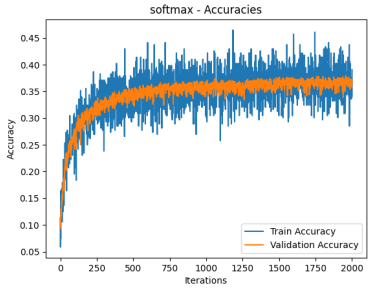
Here we are checking the last 25 validation accuracies and if the validation accuracy is not greater than at least one of the validation accuracy among the last 25 accuracies then early stopping.

Results

Using learning rate = 0.001, without any early stopping, running it for 2000 iterations, the model attained an accuracy of **32.03%**.



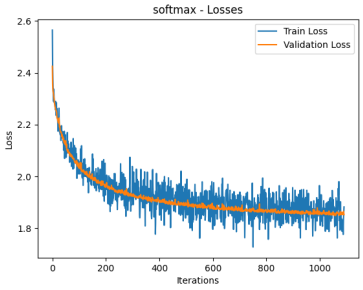
(a) Softmax loss with lr=0.001



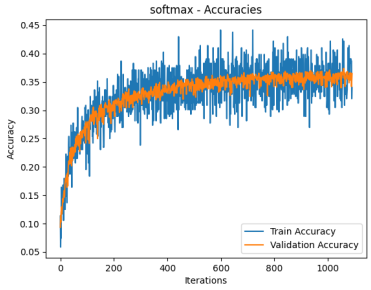
(b) Softmax accuracy with lr=0.001

Figure 1: Softmax results with learning rate=0.001

When we change the learning rate to 0.01, it is observed that convergence of the model is faster. The model attained an accuracy of **36.47%** with early stopping. This suggests that the higher learning rate facilitated quicker convergence to a better accuracy level, leading to early stopping and preventing overfitting. The early stopping occurs at 1090th iteration while running the model for 2000 iterations.



(a) Softmax loss with lr=0.01



(b) Softmax accuracy with lr=0.01

Figure 2: Softmax results with learning rate=0.01

Problem 2: Contrastive Representation Learning

1.Encoder Model

The architecture used is CNN followed by a fully connected layer. It is a Custom Convolutional Neural Network with fully connected layer for Image Feature extraction. It consists of 4 convolutional layers followed by max-pooling layers for feature extraction, and then fully connected layers for classification or further processing.

Number of units in each layer:

- First Conv2d layer: 3 input channels, 64 output channels, kernel size 3x3, padding 1
- Second Conv2d layer: 64 input channels, 64 output channels, kernel size 3x3, padding 1
- Third Conv2d layer: 64 input channels, 128 output channels, kernel size 3x3, padding 1
- Fourth Conv2d layer: 128 input channels, 128 output channels, kernel size 3x3, padding 1
- First Linear layer: 12888 input features, 4096 output features
- Second Linear layer: 4096 input features, output features

The Activation Function used is Rectified Linear Unit (ReLU). Dropout regularization with a dropout rate of 0.1 is applied after the first ReLU activation in the classifier. Batch Normalization is applied after each convolutional layer to stabilize and accelerate training. MaxPooling layers with a kernel size of 2x2 and a stride of 2 are used to downsample the feature maps spatially.

We found that using basic models like LeNet didn't give good results. And when we tried bigger models like ResNet, they caused problems because they needed too much memory on the GPU.

2. Network A Training

The cost function used is triplet margin loss with a specified distance funtion here. Used torch library to compute the specified loss. The distance function used to compute the pairwise distances between the anchor, positive, and negative samples in the triplet. it calculates the Euclidean distance between each pair of input tensors. Other parameters such as margin=1, L2 distance metric = 2.

The training is done in the following steps. The anchor, positive, and negative samples were passed into the encoder to get their vector representations. Find loss using triplet margin loss. Based on loss the encoder parameters are updated using Adam optimizer.

Parameter	Value
Learning Rate	0.001
Batch Size	256
Number of Iterations	3000

Table 2: Parameter Values for cont_rep

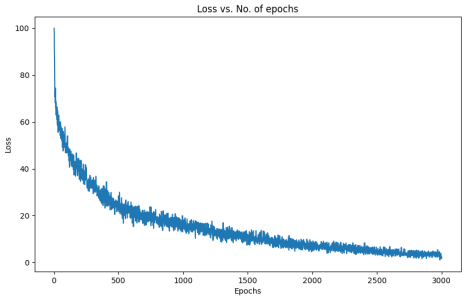


Figure 3: Loss vs iterations in the training of cont_rep

Looking at Figure 3, we notice that as the number of iterations increases, the loss decreases. This suggests that the gap between the anchor and positive samples is getting smaller, while the gap between the anchor and negative samples is getting larger.

3. t-SNE plot

Observations about t-SNE plot

Points that belong to the same digit class are likely to cluster together. Distances between points in the plot do not necessarily represent their original Euclidean distances, as t-SNE aims to preserve local structure rather than global distances. Well-separated clusters indicate that the network has learned distinct representations for different digit classes. Overlapping clusters or scattered points may indicate ambiguity in the representations learned by the network. Figure 4 represent the t-SNE plot with a margin of 50, corresponding to iterating over the entire training set and obtaining the corresponding trained vectors for all images.

Boosting the margin hyperparameter in the Triplet Margin Loss function greatly influences how the z-dimension vectors from the encoder are clustered. As this margin grows, it strongly penalizes insufficient separation between anchor-positive pairs and anchor-negative pairs. Thus, the gap between anchor and positive samples shrinks, while the gap between anchor and negative samples widens. This results in clearer groupings or clusters in the embedding space.

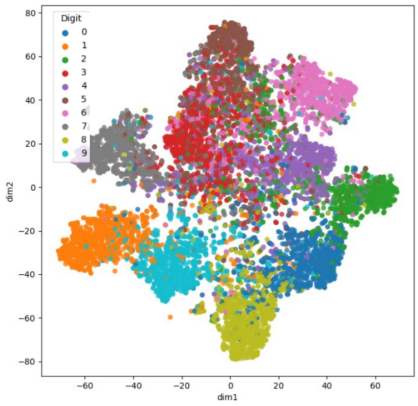


Figure 4: t-SNE plot

4. Training Linear Classifier using Learned Vectors

Parameter	**Value**
Learning Rate	0.001
Batch Size	256
L2 Regularization Lambda	0.1
Number of Iterations	2000

Table 3: Parameter Values for training Linear Classifier

Freeze the learned vectors and use them as features to train a multi-class logistic regression classifier by using the softmax regression from problem 1.

The training process includes Feature Extraction and utilizing the pre-trained model. Fine-tuning the linear classifier, learned feature vectors are then passed as input features to train a multi-class logistic regression classifier. After different combinations of hyperparameters during fine-tuning the model attains a best accuracy of **70**%.

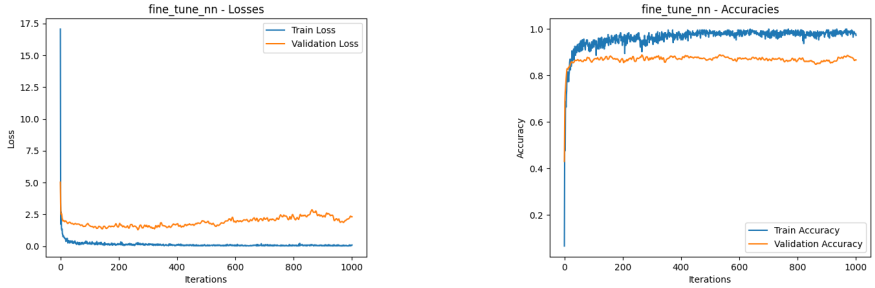
Validation process, Feature Extraction, we've encoded all examples from the validation set of CIFAR10 using the pre-trained encoder model to get the respective feature vectors in a certain dimension. Inference with Trained Classifier, Instead of training the classifier more, we're simply using the already trained linear classifier to guess labels for the validation set using the learned feature vectors. After trying different hyperparameters values got a best accuracy of **65%**.

5.Classifier Neural Network

Parameter	Value
Learning Rate	0.001
Batch Size	256
Number of Iterations	1000

Table 4: Parameter Values

Training the network B using the negative log likelihood cost function on examples from the training set. we are getting a classification accuracy of Network B on the training and validation sets 98% and 88.7% respectively.



(a) fine_tune_nn loss

(b) fine_tune_nn accuracy

Figure 5: fine_tune_nn results

In most cases, fine-tuning (Problem 2.5) should lead to better results compared to using a single-layer logistic regression classifier (Problem 2.4).

Network A has already learned valuable features from the general-purpose dataset. These features can be beneficial for digit classification as well, providing a strong foundation for Network B.

By starting with pre-trained weights, Network B requires less training data and time to achieve good performance. This is especially helpful when dealing with limited datasets.

Transferring knowledge from a larger dataset can lead to a more robust model that generalizes better to unseen data.

But, in some cases single-layer logistic regression might outperform. If the digit classification task is very basic, a logistic regression model might be sufficient and simpler to train and If the training data is extremely limited, the additional complexity of Network B might not be beneficial.

I got the best score of **82.41** in the leaderboard corresponding to SR number 22523.