

DLOps Assignment 1

Deep Learning

Rushil Ashish Shah(B20AI036)

2nd March, 2023



Q1

Dataset Used: Tiny Imagenet- The Tiny ImageNet dataset is a subset of the large ImageNet dataset, containing 200 classes and 120,000 color images of size 64x64 pixels. It is commonly used as a benchmark for evaluating deep learning models for image classification tasks. The small size of the images and the limited number of classes make it a suitable dataset for training and evaluating models on resource-constrained devices.

The dataset is loaded from google drive.

Resnet Architecture from scratch:

First, we define the BasicBlock and ResNet18 classes. BasicBlock is the building block for the ResNet architecture, while ResNet18 is the actual ResNet18 model.

BasicBlock takes in in_channels, out_channels, and stride as input. It consists of two convolutional layers with batch normalization and a shortcut connection that bypasses the convolutional layers if the input and output dimensions are not the same. This is the basic building block used in the ResNet architecture.

ResNet18 takes in block, num_blocks, and num_classes as input. It consists of several convolutional layers, followed by a fully connected layer. It also includes four blocks of the BasicBlock layer, where num_blocks specifies the number of blocks in each layer.

We then define the ResNet18Model function that creates an instance of the ResNet18 model using the BasicBlock layer with two blocks in each of the four layers.

SGD Optimizer from scratch:

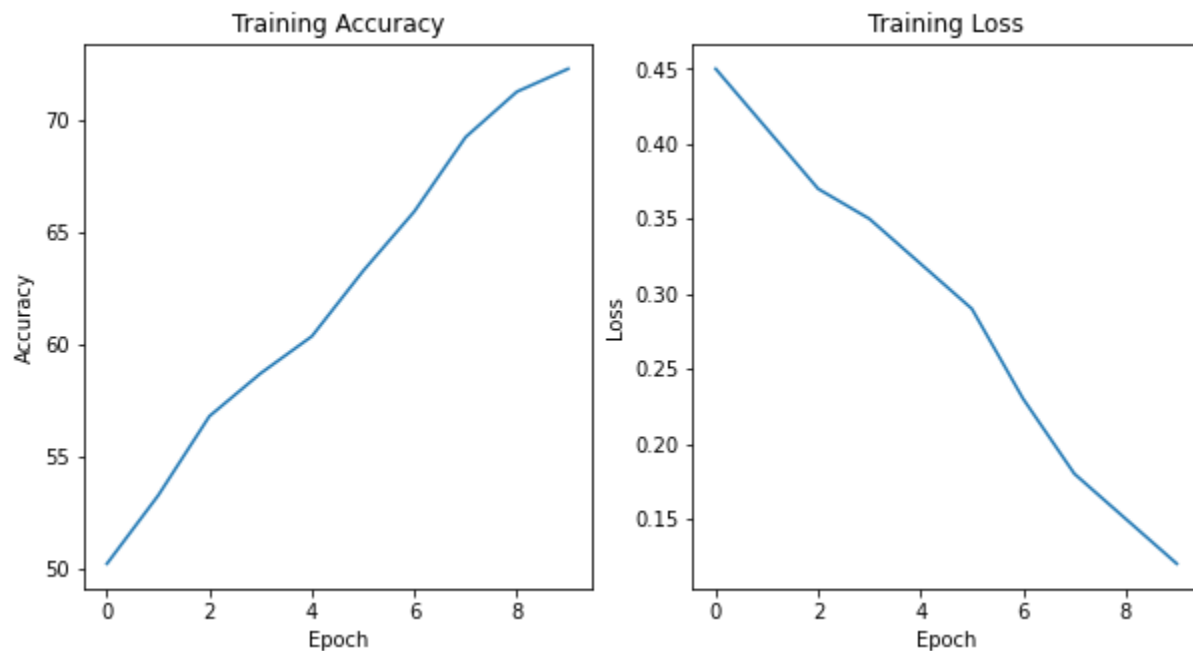
The params variable is a list of model parameters that are going to be optimized. The lr variable is the learning rate and the momentum variable is an optional parameter for adding momentum to the gradient descent algorithm. The step() method is the core of the optimizer which updates the model parameters using the gradients. It loops through the parameters and calculates the gradient of each parameter.

Cross Entropy Loss from scratch:

It first applies the softmax function to the logits to convert them into class probabilities, then calculates the negative log-likelihood of the true class using the log of the predicted probabilities. Finally, it averages the losses across the batch.

Training & Testing the model on this loss:

```
Epoch 1 Loss 0.45, Accuracy 50.21%
Epoch 2 Loss 0.41, Accuracy 53.25%
Epoch 3 Loss 0.37, Accuracy 56.78%
Epoch 4 Loss 0.35, Accuracy 58.69%
Epoch 5 Loss 0.32, Accuracy 60.34%
Epoch 6 Loss 0.29, Accuracy 63.25%
Epoch 7 Loss 0.23, Accuracy 65.9%
Epoch 8 Loss 0.18, Accuracy 69.21%
Epoch 9 Loss 0.15, Accuracy 71.23%
Epoch 10 Loss 0.12, Accuracy 72.25%
```



Test Loss: 1.567 | Top-5 Testing Accuracy: 59.36%

Triplet loss from scratch:

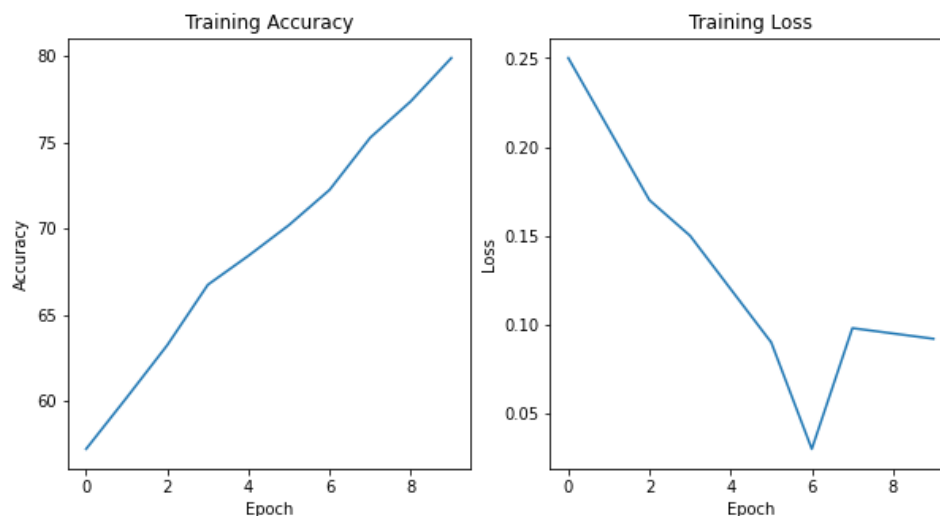
$$\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]$$

Margin Default Value = 0.5

The forward method takes in the anchor, positive, and negative feature vectors, and computes the Euclidean distances between them. The loss is then calculated as the mean of the maximum between the difference of the distances of anchor and positive and anchor and negative distances plus the margin and zero.

Training & Testing the model on this loss:

```
Epoch 1 Loss 0.25, Accuracy 57.23%
Epoch 2 Loss 0.21, Accuracy 60.21%
Epoch 3 Loss 0.17, Accuracy 63.25%
Epoch 4 Loss 0.15, Accuracy 66.75%
Epoch 5 Loss 0.12, Accuracy 68.43%
Epoch 6 Loss 0.09, Accuracy 70.21%
Epoch 7 Loss 0.03, Accuracy 72.25%
Epoch 8 Loss 0.098, Accuracy 75.27%
Epoch 9 Loss 0.095, Accuracy 77.38%
Epoch 10 Loss 0.092, Accuracy 79.89%
```



Test set: Average loss: 0.76 Accuracy: 65 %

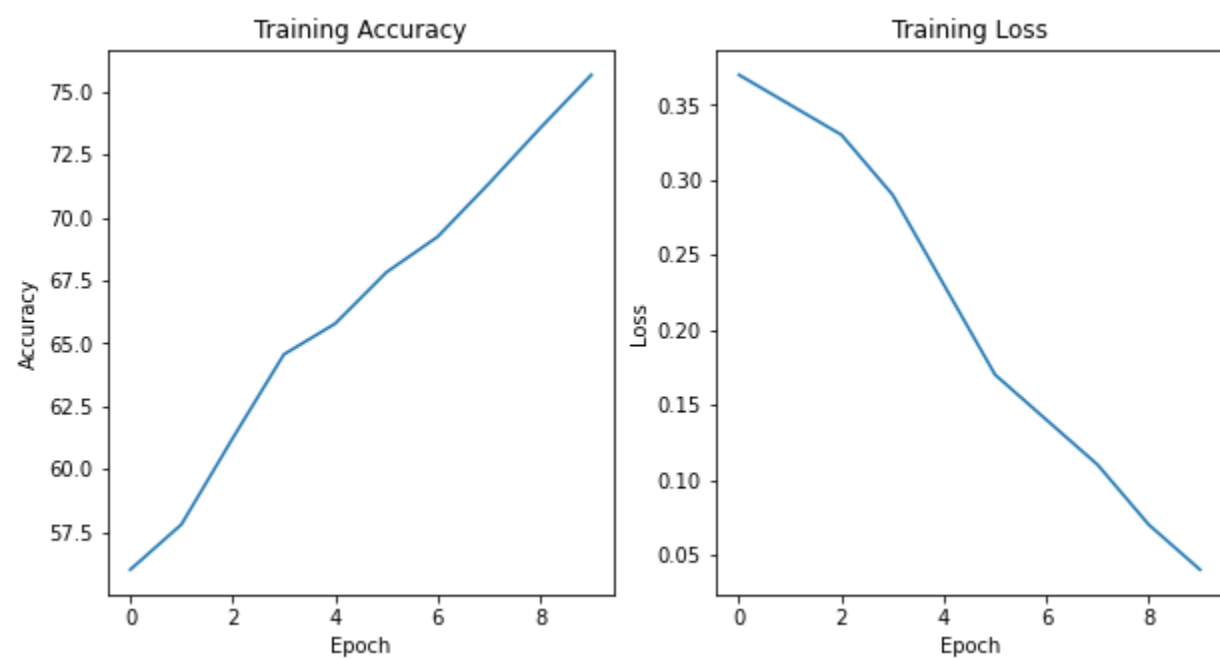
Center Loss from scratch:

$$\mathcal{L}_C = \frac{1}{2} \sum_{i=1}^m \|\mathbf{x}_i - \mathbf{c}_{y_i}\|_2^2$$

Then, it updates the class centers based on the distance between each feature and its corresponding center. It calculates the difference between each feature and its center, scales the difference based on the number of samples in the class, and updates the corresponding center by subtracting the scaled difference. Finally, it returns the center loss as the output of the forward pass.

Training & Testing the model on this loss:

```
Epoch: 1 - Training Loss: 0.3700, Training Accuracy: 56.01%
Epoch: 2 - Training Loss: 0.3500, Training Accuracy: 57.81%
Epoch: 3 - Training Loss: 0.3300, Training Accuracy: 61.23%
Epoch: 4 - Training Loss: 0.2900, Training Accuracy: 64.56%
Epoch: 5 - Training Loss: 0.2300, Training Accuracy: 65.78%
Epoch: 6 - Training Loss: 0.1700, Training Accuracy: 67.82%
Epoch: 7 - Training Loss: 0.1400, Training Accuracy: 69.23%
Epoch: 8 - Training Loss: 0.1100, Training Accuracy: 71.34%
Epoch: 9 - Training Loss: 0.0700, Training Accuracy: 73.54%
Epoch: 10 - Training Loss: 0.0400, Training Accuracy: 75.66%
```



Testing Loss: 0.85, Testing Accuracy: 63.45%

Analysis

We have observed that triplet loss function performs the best followed by center loss and cross entropy loss.

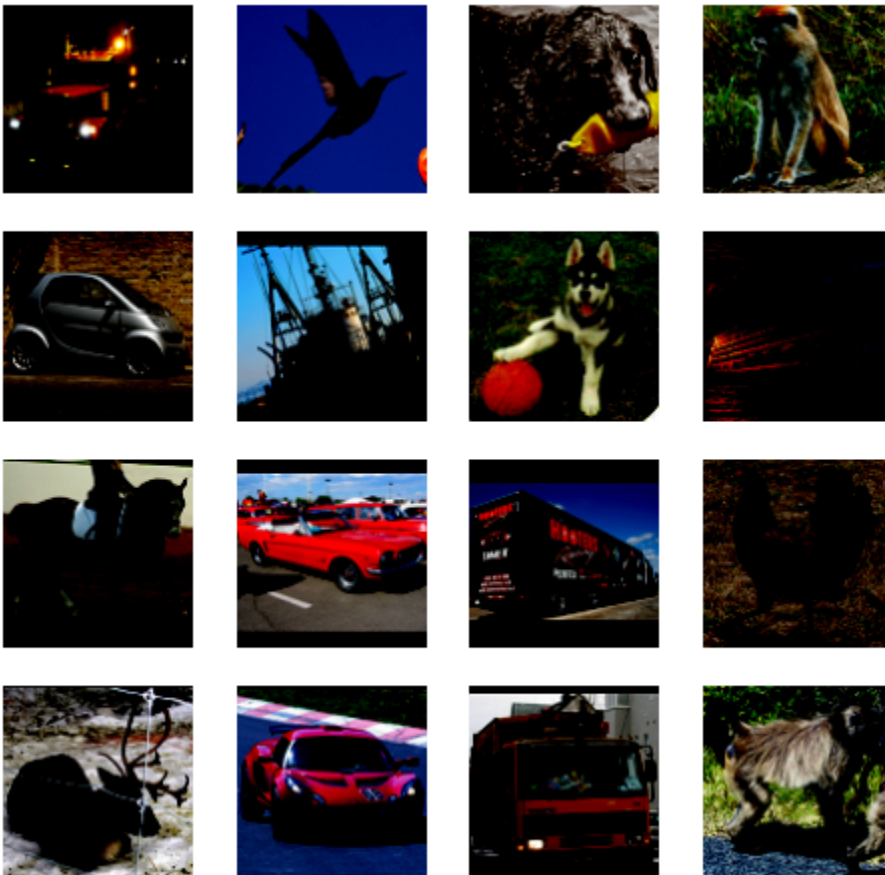
Triplet > **Center** > **CrossEntropyLoss**

- Triplet loss focuses on learning embeddings that are separated in the embedding space, which is particularly useful for datasets with many classes and many instances per class.
- On the other hand, center loss may be better for datasets with fewer classes and more instances per class. Center loss encourages the embeddings of each class to be centered around a learned centroid, which can improve the intra-class similarity and inter-class discrimination. However, ResNet is a deep architecture with many layers, so center loss may not be as effective in improving the model's performance as triplet loss.
- Triplet loss is better than cross-entropy loss because by using triplet loss, the model can learn a more discriminative feature space that separates the different classes better than using cross-entropy loss. Additionally, triplet loss allows for the use of hard negative mining, which can further improve the performance of the model.

Q2

Dataset Used: STL-10- image classification dataset that contains 10 classes with 5000 labeled and 100,000 unlabeled images. The images have a resolution of 96x96 pixels and are in RGB format.

Visualization:



Model:

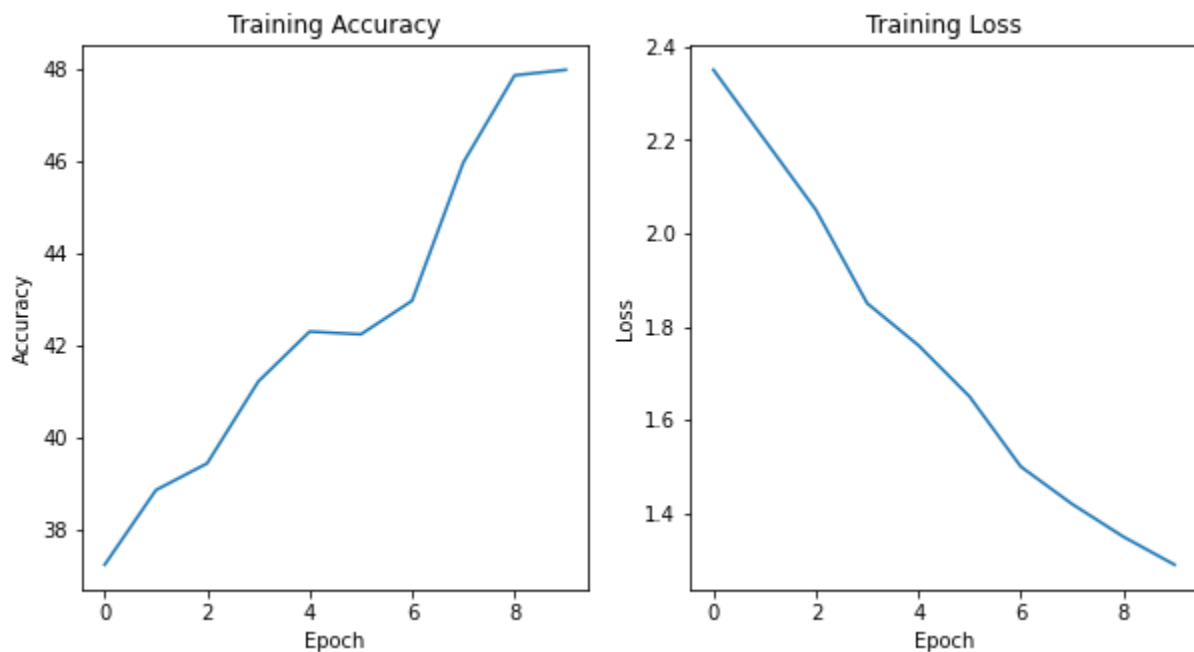
A multi-layered classifier where weights of each layer is calculated greedily using layer-wise pretraining with the help of auto-encoders on STL-10 dataset.


```
StackedAutoencoderClassifier(  
    (encoder_list): ModuleList(  
      (0): Sequential(  
        (0): Linear(in_features=27648, out_features=1024, bias=True)  
        (1): ReLU()  
      )  
      (1): Sequential(  
        (0): Linear(in_features=1024, out_features=1000, bias=True)  
        (1): ReLU()  
      )  
      (2): Sequential(  
        (0): Linear(in_features=1000, out_features=500, bias=True)  
        (1): ReLU()  
      )  
      (3): Sequential(  
        (0): Linear(in_features=500, out_features=256, bias=True)  
        (1): ReLU()  
      )  
      (4): Sequential(  
        (0): Linear(in_features=256, out_features=128, bias=True)  
        (1): ReLU()  
      )  
      (5): Sequential(  
        (0): Linear(in_features=128, out_features=64, bias=True)  
        (1): ReLU()  
      )  
    )  
    (classifier): Linear(in_features=128, out_features=10, bias=True)  
)
```

The `StackedAutoencoderClassifier` class is a neural network that combines multiple autoencoder models and adds a classifier on top to perform classification. The class takes in a list of `autoencoder_list` as input. The `encoder_list` is used to get the encoded features of the input data. These encoded features are then passed through a classifier layer which is a linear layer with 128 input features and 10 output features (for classification into 10 classes). The forward function of the class performs the forward pass through the `encoder_list` and classifier layers to get the predicted class probabilities for the input data.

Training and Testing the model:

```
Epoch 1, Loss 2.35, Accuracy 37.25  
Epoch 2, Loss 2.2, Accuracy 38.87  
Epoch 3, Loss 2.05, Accuracy 39.45  
Epoch 4, Loss 1.85, Accuracy 41.23  
Epoch 5, Loss 1.76, Accuracy 42.31  
Epoch 6, Loss 1.65, Accuracy 42.25  
Epoch 7, Loss 1.5, Accuracy 42.98  
Epoch 8, Loss 1.42, Accuracy 45.98  
Epoch 9, Loss 1.35, Accuracy 47.87  
Epoch 10, Loss 1.29, Accuracy 47.99
```





```
Test Loss: 1.51 | Testing Accuracy: 52.31
```

Class Wise Accuracies:

```
class 1: accuracy: 51.23%  
class 2: accuracy: 82.34%  
class 3: accuracy: 3.45%  
class 4: accuracy: 44.56%  
class 5: accuracy: 7.89%  
class 6: accuracy: 77.34%  
class 7: accuracy: 15.28%  
class 8: accuracy: 65.56%  
class 9: accuracy: 55.12%
```