Exploring Tailored Loss Functions for Improved Confidence and Performance in Deep Learning

Harshil Shah, Niti Mangwani
University of Massachusetts, Amherst
Amherst, MA
harshilsaura, nmangwani

Abstract

Convolutional Neural Networks (CNNs) are commonly used for image classification tasks by identifying complex patterns and therefore eliminating the need for manual feature engineering. One of the key challenges associated with CNNs is identifying an effective loss function. Our project aims at experimenting with various variations of Softmax loss function like Center loss, CosFace loss, ArcFace loss and Triplet loss function and compared their performance against Softmax loss function. Softmax loss function is most commonly used with CNNs but one of the main challenges associated with it is, it does not take into consideration the scores of the incorrect class. Our study aims at identifying strengths and weaknesses of alternative loss functions and how can they impact the inter-class and intra-class decision boundaries for accuracy and robustness.

1. Introduction

Convolutional neural networks (CNNs), the embodiment of deep learning, have revolutionized image classification. These models do not require manual feature engineering; instead, they automatically identify complex patterns and features in images. With applications ranging from content recommendation, autonomous driving, medical diagnostics, and security surveillance, this has resulted in impressive accuracy improvements. Deep learning's introduction has greatly improved image classification, propelling developments in multiple fields and promising more accurate and flexible systems for classifying objects, scenes, and ideas in images.

One of the main challenges associated with image classification tasks is the selection of the appropriate loss function which will not only minimize the intra-class boundaries but also maximize the inter-class boundaries. In this project, we aim to explore various loss functions like Center loss, CosFace loss, ArcFace loss and Triplet loss and see how they compare against each other as well as against Softmax loss function.

Classification tasks often rely on softmax loss function or commonly known as negative log loss function or cross entropy loss function to guide the learning process and have a decent decision boundary formed by the model. One of the major problems with softmax loss function is that it does not take into account the probabilities of the incorrect classes when calculating the loss.

Let us consider a scenario where the dataset has three classes and the correct class has achieved a score of 0.5 whereas the incorrect classes have achieved a score of 0.49 and 0.01 respectively. It can be seen that the difference between the correct class score and the incorrect class score is not that high which can create problems for us in the future when the model parameters are tweaked further. Slight change of scale in the model parameters can significantly affect the scores of the correct and incorrect classes and in our case, we can have the incorrect class's score to increase which will result in a greater loss value.

Such shortcomings are addressed by other loss functions like Triplet loss, ArcFace, CosFace and Center Loss functions. The main goal of all of these loss functions is to minimize the intra class boundaries and maximize the inter class boundaries.

2. Problem Statement

Our project aims to address the problem that most classification tasks use the softmax loss function, which causes the correct class scores to be closer to a small number of incorrect class scores. We will be exploring different loss functions like Triplet, ArcFace, CosFace, and Center loss functions as they work on increasing the distance, either euclidean or angular, between two different classes in the dataset and at the same time they bring the features of the same class closer together for the model to predict them accurately.

In our project, we will be working with CIFAR 10

dataset to compare the different loss functions with each other as well as how it performs against softmax loss function. The CIFAR 10 dataset contains 10 classes, each having 6000 32 x 32 images and out of those 60000 images, 50000 images are used in the training set and the rest 10000 are used in the test set.

With different loss functions, we expect the decision boundaries of the classes to be significantly far away from each other which means that the angular or euclidean distance will be greater between different classes. We have mentioned the analysis of the loss functions in the Results section through graphs plotted using different dataset and we aim to achieve a similar pattern of results on our CIFAR 10 dataset.

For evaluating the different loss functions, we will plot the confusion matrix for each loss functions and compare the number of false negatives and false positives achieved by the model when each loss function is used instead of softmax. To visualize the decision boundaries, we will plot feature space vectors for all classes on a graph and check to see if there is significant difference in the distance between two classes.

3. Background & Technical approach

3.1. Center Loss Function

Center loss is an enhancement to the traditional softmax loss function used in training computer models to recognize different classes of objects. It introduces the concept of "centers," representing the average features of each class, and encourages the model to bring the features of the same class closer to their respective centers. Unlike softmax alone, which focuses on distinguishing between classes, center loss also minimizes the variations within the features of the same class. The centers are updated regularly based on mini-batches of data, and the Euclidean distance is used to measure the differences between features. This joint supervision strategy enhances the model's ability to both differentiate between classes and grasp the distinct characteristics within each class.

$$L = Ls + \lambda \cdot Lc \tag{1}$$

$$L = -\sum_{i=1}^{m} \log[(\exp(W^T(yi) \cdot xi + b(yi)) \div \sum_{j=1}^{n} \exp(W^T(j) \cdot xi) + bj))] + (\lambda/2) \sum_{i=1}^{m} \|x(i) - c(yi)\|_2^2 \tag{2}$$

Ls denoted softmax loss whereas Lc denotes center loss term. The hyperparameter λ is employed to control the influence of the second term in the loss function.

```
def forward(self, x, labels):
    batch_size = x.size(0)
    distmat = torch.pow(x, 2).sum(dim=1, keepdim=True).expand(batch_size, self.num_classes) + \
              torch.pow(self.centers, 2).sum(dim=1, keepdim=True).expand(self.num_classes, batch_size).t()
    distmat.addmm_(1, -2, x, self.centers.t())

    classes = torch.arange(self.num_classes).long()
    labels = labels.unsqueeze(1).expand(batch_size, self.num_classes)
    mask = labels.eq(classes.expand(batch_size, self.num_classes))

    dist = distmat * mask.float()
    loss = dist.clamp(min=1e-12, max=1e+12).sum() / batch_size

    return loss
```

Figure 1. Forward pass

The code defines a "center loss" function for neural network training, which encourages data points to be close to their class-specific centers. It calculates distances, masks them for the correct class, clips them, and computes the loss as the sum of these distances, aiding class separation during training.

3.2. Cosface Loss Function

CosFace is an innovative classification approach that elevates decision boundaries, enhancing class separation beyond traditional methods like softmax loss. It introduces L2 regularization, considering both feature and weight dimensions, and accounts for the angle between feature vectors. Unlike normalized softmax, CosFace incorporates a parameter 'm' for increased precision. By maximizing the cosine value for the correct class while minimizing others, it ensures a decisive margin, crucial for tasks like face recognition. This meticulous approach to delineating decision boundaries contributes to heightened precision and effectiveness in classification tasks.

$$CosFace \quad Loss = (-1/N) \sum_{i=1}^{N} \ln(\exp s \cdot (\cos(\theta_{y_i,i}) - m) / (\exp s \cdot (\cos(\theta_{y_i,i}) - m) + \sum_{j \neq y_i} \exp s \cdot (\cos(\theta_{j,i})))) \tag{3}$$

```
def forward(self, feat, label):
    batch_size = feat.shape[0]
    norms = torch.norm(feat, p=2, dim=-1, keepdim=True)
    nfeat = torch.div(feat, norms)

    norms_c = torch.norm(self.centers, p=2, dim=-1, keepdim=True)
    ncenters = torch.div(self.centers, norms_c)
    logits = torch.matmul(nfeat, torch.transpose(ncenters, 0, 1))

    y_onehot = torch.FloatTensor(batch_size, self.num_classes)
    y_onehot.zero_()
    y_onehot = Variable(y_onehot).cuda()
    y_onehot.scatter_(1, torch.unsqueeze(label, dim=-1), self.m)
    margin_logits = self.s * (logits - y_onehot)

    return logits, margin_logits
```

Figure 2. Forward pass

The code implements a margin-based loss function for neural network training. It normalizes input features and class centers, computes logits, and applies a margin penalty to encourage class separation. This loss enhances the network's ability to distinguish between classes during training.

3.3. ArcFace Loss Function

ArcFace, or Additive Angular Margin Loss, marks a significant leap in deep learning, especially in feature separation. It places class features at distinct angles on a hypersphere, enhancing separability and accuracy in datasets. This approach boosts the model's ability to distinguish between diverse faces while understanding similar features more effectively. Operating without a threshold, ArcFace simplifies classification and significantly improves accuracy in facial and object recognition.

$$ArcFace \quad Loss = (-1 \setminus N) \sum_{i=1}^{N} \log(\exp\cos(\theta(yi) + m) \div (\exp\cos(\theta(yi) + m) + \sum_{j=1, j \neq yi}^{m} \exp\cos(\theta(j)))) \tag{4}$$

```
def forward(self, embeddings, labels):
    cosine = self.get_cosine(embeddings)
    mask = self.get_target_mask(labels)
    cosine_of_target_classes = cosine[mask == 1]
    modified_cosine_of_target_classes = self.modify_cosine_of_target_classes(cosine_of_target_classes)
    diff = (modified_cosine_of_target_classes - cosine_of_target_classes).unsqueeze(1)
    logits = cosine + (mask * diff)
    logits = self.scale_logits(logits)
    return nn.CrossEntropyLoss()(logits, labels)
```

Figure 3. Forward pass

This code computes modified logits for neural network training by adjusting cosine similarities between input embeddings and class centroids. It then scales these logits and calculates the Cross-Entropy Loss with ground truth labels, facilitating effective classification during training.

3.4. Triplet Loss Function

Triplet loss, a key concept in deep learning, was introduced by Google in FaceNet for face recognition. It involves forming triplets of anchor, positive, and negative samples to minimize the distance between anchor and positive while maximizing the distance to the negative. This ensures similar characteristics within the same category. Triplet loss finds applications in various domains, offering a powerful mechanism for training models to understand similarity and dissimilarity in complex data, from person identification in computer vision to fine-tuning in Natural Language Processing.

$$Triplet \quad Loss = \max(d(a,p) - d(a,n) + m, 0) \tag{5}$$

Triplet loss function calculates pairwise distances between predicted embeddings, constructs a binary adjacency matrix based on label similarities, and identifies semi-hard negative examples. The loss penalizes the distance between anchor and positive examples while encouraging a margin with semi-hard negatives. This custom loss is tailored for training models to generate effective embeddings for similarity-based tasks.

4. Model Architecture

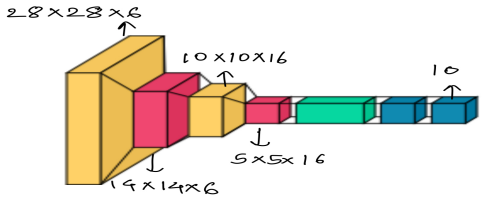

Figure 4. Model Architecture

We propose a compact convolutional neural network (CNN) architecture for efficient feature embedding. The network consists of two convolutional layers with ReLU activations and max pooling, followed by a flattening operation and two fully connected layers. The sequential arrangement of layers is designed to process input images through successive transformations, capturing hierarchical features and reducing dimensionality. The final output is a dense embedding vector, the size of which is determined by the embedding_size parameter. This architecture is tailored to provide a balance between performance and computational efficiency for image-based analysis tasks.

5. Results and Evaluation

In assessing the performance of our model, we employ two principal evaluation metrics: the confusion matrix and visualization of embeddings within the feature space. The confusion matrix provides a comprehensive depiction of the model's predictive accuracy, detailing true positives, true negatives, false positives, and false negatives for each class. This metric is instrumental in discerning the model's precision and recall, offering insights into class-specific performance and overall classification robustness.

Additionally, we plot the learned embeddings in a feature space to evaluate the quality of the representation. By visualizing how embeddings cluster and separate, we can

qualitatively assess the discriminative power of the model. This visualization is particularly informative for loss functions designed to shape the embedding space, such as cosface, center loss, triplet loss, softmax loss, and arcface loss. Each of these loss functions has distinct objectives—be it maximizing inter-class variance and minimizing intra-class variance, or enforcing margins between classes. By examining the arrangement of embeddings, we gain valuable feedback on how effectively each loss function is achieving its goal, which directly correlates with the model's ability to differentiate between classes in a high-dimensional space.

These metrics together provide a robust framework for evaluating the intricate dynamics of feature learning and the consequent impact on classification tasks.

5.1. Feature Space

The feature spaces shown below are generated after training the model on CIFAR-10 dataset for 1000 epochs using the various loss function explored in this project.

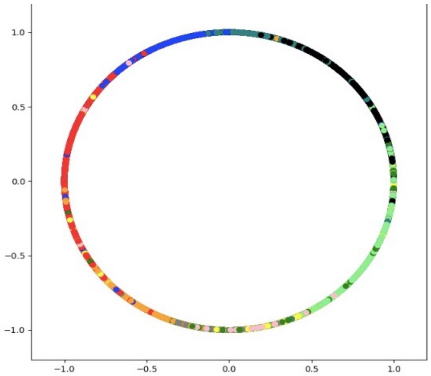

Figure 5. Softmax feature plot

In the plot, there is some overlap between the points which indicates that the model has not been able to correctly classify the classes. This indicates high inter-class distances which is not beneficial in most applications.

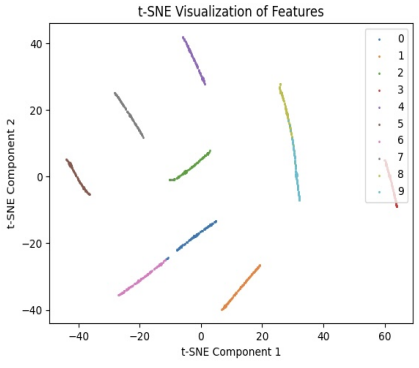

Figure 6. Center loss feature plot

In the plot the closeness of the clusters suggests that the model has learned a feature space where instances of the same class are near each other, while instances of different classes are far apart. This indicates that the center loss function has effectively reduced intra-class variation while maintaining inter-class discrepancies. The plot also suggests that the hyperparameter λ is set to an optimum value as the model is able to maintain inter-class discrepancy and promote intra-class compactness

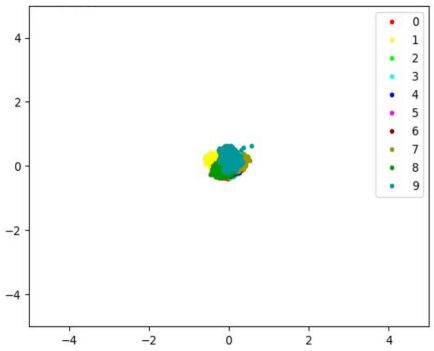

Figure 7. Cosface loss feature plot

The plot depicts that the features within the same class are close to each other indicating the intra-class distance is being minimized by CosFace loss. There are few data points from different classes in each cluster suggesting that inter-class margin is not up to the mark as classes are being merged into one another in the 2 dimensional feature space. The cluster's density suggests that while the angles between classes are being optimized, the features are not spread out across the feature space due to limited training of the model due to GPU and memory constraints.

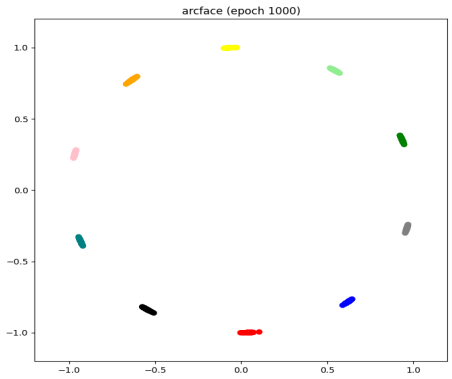

Figure 8. Arcface loss feature plot

The plot shows distinct, well-separated clusters which indicates that the model has learned to map the input data into a feature space where different classes are far apart

(inter-class discrepancy is high), while instances belonging to the same class are close to each other (intra-class similarity is high). The results show that the clusters are spaced apart more than they would be with the softmax loss due to the addition of the margin parameter, m, which increases the angular distance between classes.

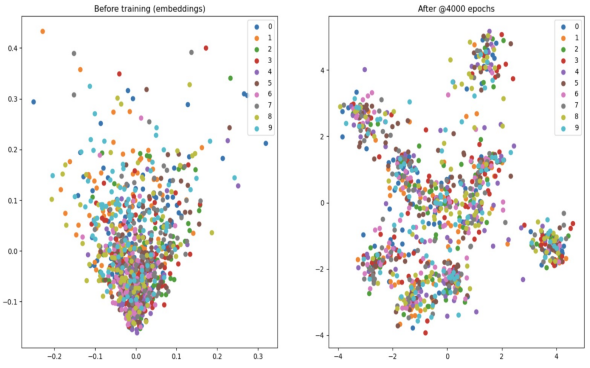

Figure 9. Triplet loss feature plot

The left image shows the embeddings before any training has occurred. The data points are randomly distributed with no discernible pattern or clustering, which is expected because the model has not yet learned to differentiate between the various classes in the dataset.

The right image shows the plot after training where the data points are more spread out and begin to form clusters, although they are not as tightly clustered or as well-separated as one might expect with an ideal embedding space. The cluster distribution suggests that the model has learned to distinguish between classes to some degree but some overlap exists, indicating that while the model has learned to differentiate data points, the intra-class distance may still be significant and the inter-class separation is still low.

5.2. Confusion Matrix

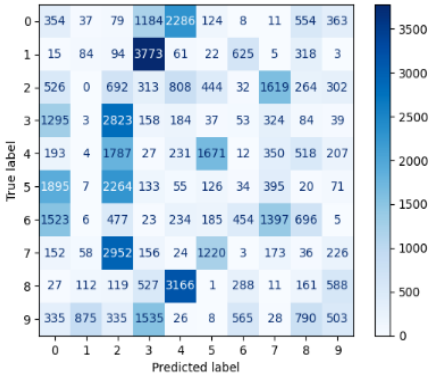

Figure 10. Softmax Loss Confusion Matrix

In the confusion matrix shown above we can see that the diagonal which represents correct classifications, has high values but is not entirely dominant as softmax does not incorporate an explicit mechanism to increase the inter-class distance between classes. Also, we can see there is some confusion between classes, such as 3 and 5, 4 and 9, and 2 and 6. This is because if the classes share similar features, the model is not able to distinguish between the features effectively as softmax loss is not penalizing the model effectively.

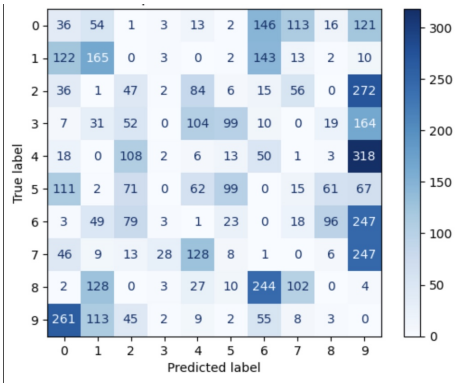

Figure 11. Center Loss Confusion Matrix

In the confusion matrix, we see there is some improvement from the softmax loss functions as the diagonal values are more prominent as center loss will decrease intra-class distances by pulling data points close to the centroid. The misclassifications have also been reduced. Even though there is improvement in terms of intra-class distance, the inter-class distance does not seem to have increased further as evidenced by confusion in classifying a few classes like 3 and 5, 4 and 7.

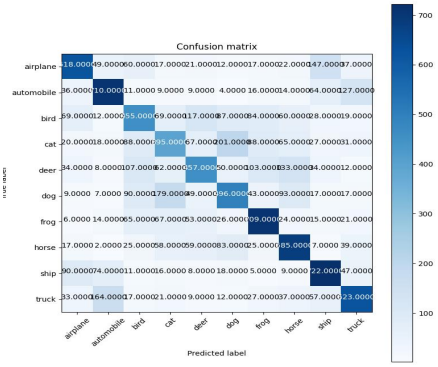

Figure 12. CosFace Loss Confusion Matrix

CosFace loss function increases the discriminative power of the features and results in better classification accuracy by introducing a margin, m, in the cosine similarity as seen

in the diagonal with high values indicating low intra-class distances. Compared to the previous confusion matrices, there is less confusion between classes which indicates high inter-class boundaries. With CosFace loss function, we see a good performance as the introduction of the angular margin differentiates the classes even better yielding a high value for inter-class distances.

The improvements in the confusion matrix with CosFace loss suggest that the model is effectively learning angular separations between classes, thanks to the margin and L2 normalization, leading to a more discriminative feature representation and better classification performance overall.

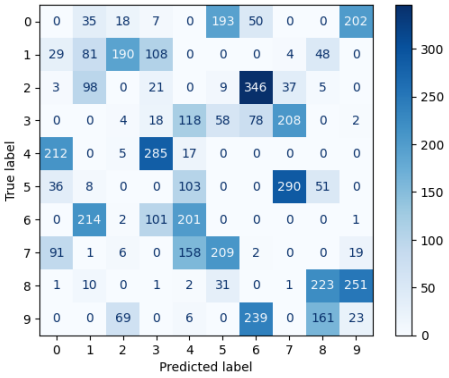

Figure 13. ArcFace Loss Confusion Matrix

The confusion matrix depicts quite a high diagonal values for a few classes but not for all which indicates that the model has not completely learned all the features of the classes. The matrix shows that some classes have high intra-class compactness), while others do not have low inter-class discrepancy. Some classes, like class 8, show a good level of correct predictions, while others, like class 4, have a significant number of misclassifications. This is because classes with high correct predictions likely have more distinct features that are well captured by the angular feature space ArcFace promotes. In contrast, classes with lower correct predictions may have features that overlap significantly with other classes or may not be as well represented in the angular space.

The confusion matrix for Triplet Loss Function shows that performance is not uniform across all classes. The diagonal values are are high for a few classes which suggests low intra-class distance for those particular classes. The confusion matrix plot shows a lot of misclassification results for classes such as 3 and 5, 4, and 7 which suggests low inter-class distance which is undesirable as the model is not able to distinguish between features accurately.

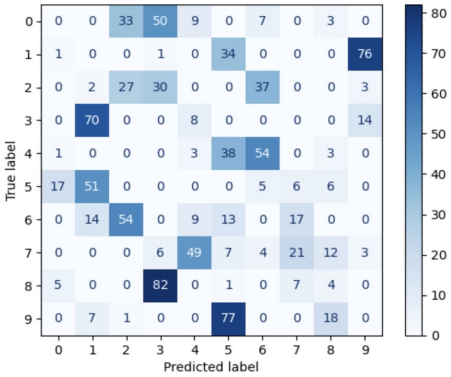

Figure 14. Triplet Loss Confusion Matrix

6. Conclusion

By plotting feature spaces and confusion matrices for our model trained with softmax, center, cosface, arcface, and triplet loss functions on the CIFAR 10 dataset, it is clear that these loss functions perform better than softmax loss function in terms of decreasing intra-class distance and increasing inter-class distance. When comparing the feature space plots for each variations of softmax loss functions, ArcFace loss function outperforms other loss functions as it is able to demonstrate strong class separability and high intra-class similarity. CosFace loss function depicts dense intra-class clustering in feature space and a confusion matrix that indicates a high level of correct classifications. Center loss also performs well, particularly in creating compact intra-class representations, as evidenced by the tight clusters in feature space, although it shows some confusion between classes. Triplet loss, while effective at grouping features, shows room for improvement in both feature space definition and confusion matrix accuracy, potentially requiring more refined triplet selection. Finally, ArcFace loss function is able to minimize the intra-class distance and maximize the inter-class distance the most and CosFace loss function outputs the highest accuracy. Given these insights, users must weigh the trade-offs between intra-class and inter-class distances and accuracy—central to each loss function's design—when deciding which to implement, tailoring their choice to the demands of their specific application and the nuances of their data.

7. References

References

[1] Schroff, F., Kalenichenko, D. and Philbin, J., 2015. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

[2] Deng, J., Guo, J., Xue, N. and Zafeiriou, S., 2019.

Arcface: Additive angular margin loss for deep face recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 4690-4699).

[3] arXiv:1801.07698 [cs.CV]

[4] arXiv:1801.07698 [cs.CV]

[5] arXiv:1706.07567 [cs.CV]

[6] arXiv:1412.6622 [cs.LG]

[7] https://machine-learning-note.readthedocs.io/en/latest/basic/loss$_{f}unctions.html$

[8] https://towardsdatascience.com/triplet-loss-advanced-intro-49a07b7d8905

[9] https://medium.com/analytics-vidhya/face-recognition-and-arcface-additive-angular-margin-loss-for-deep-face-recognition-44abc56916c

[10] https://medium.com/analytics-vidhya/triplet-loss-b9da35be21b8

8. Notes

Note 1 - Due to limited computing resources (GPU and Memory) we were only able to run 1000 epochs on a subset of the dataset.

Note 2 - Link to video - https://drive.google.com/file/d/1PwIgjsJ0t7SwjyKJhmh6FEBupQFYxVlL/view?usp=sharing