Project 1

MSDA3100 : Applied Deep Learning

Session: Jan 2025 – May 2025

Professor – Ahmed Elsayed

Submitted by Group7

Nithin Rachakonda - C70313368

Sai Nithisha Marripelly – C70313009

Gautam Mehta – C70304767

On

16th March 2025

School of Professional Studies
Clark University

Understanding The AdaFace Paper

To understand the importance of the Adaloss we need to compare its mechanism with the traditional Cross entropy and Arcface loss functions.

In the traditional cross entropy loss, the SoftMax function focuses on just correctly classifying the samples and does not ensure similar faces are close together in the feature space. Whereas Arcface converts classification to angular classification problem. It uses a margin penalty to make decision boundaries stricter.

In Arcface, we find the angle between the normalized feature embedding and normalized class weight vector of the correct class and we add a margin to it. This makes the correctly identified samples move away from the decision boundary and come closer to the class weight vector. This forces the model to learn more distinct features for the correct class. An important thing to note here is that the margin (m) added to the cosine similarity here is an angular margin, i.e., it is added as shown below.

$$L = -\log \frac{\exp\left(s\cos\left(\theta_j + m\right)\right)}{\exp\left(s\cos\left(\theta_j + m\right)\right) + \sum_{j \neq y_i}^{n} \left(\exp(s\cos\theta)\right)}$$

This makes the loss function put more emphasis on the samples which are wrongly classified and near the decision boundary and put less emphasis on the samples that are wrongly classified and are far from the decision boundary. By enforcing this mechanism, Arcface prevents the model from giving too much importance to extremely hard-to-classify samples, which could otherwise degrade the features learned from easier to classify samples.

This is a good approach, but AdaFace brings another parameter into the picture. It argues that this approach might be good for High quality images but not for Low quality images.

Authors of Adaface argue that feature norms can be used as a proxy for image quality and they back their statement by showing 0.55 correlation between image quality and its respective feature norm. It introduces 2 margins, an additive margin (g2) and an angular margin (g1). These are introduced in the loss function as below

$$L = -\log \frac{\exp\left(s\cos\left(\theta_j + \mathrm{g1}\right) - \mathrm{g2}\right)}{\exp\left(s\cos\left(\theta_j + g1\right) - \mathrm{g2}\right) + \sum_{j \neq y_i}^{n} \exp\left(s\cos\theta_j\right)}$$

These newly defined margins are calculated using the standardized feature norms (proxy for image quality). Additionally, while calculating the normalized feature norms they clip it between -1 and 1 to stop gradient from flowing. They also used exponential moving averages for mean and standard deviation for normalizing the feature norms. They used this approach because, for small batch sizes, batch statistics can be unstable. For high batch sizes this step can be skipped as it will not be critical. However, authors recommend using this method regardless of the batch size. So, in this project we are using exponential moving averages for normalizing the feature norms as we want to implement the Adaloss identically to the implementation done by the authors. The normalized feature norm is denoted by $\|\hat{z}_i\|$. Below are the equations on how the angular and additive margins are defined:

$$g_1 = -m\|\hat{z}_i\| \quad ; \quad g_2 = m\|\hat{z}_i\| + m$$

This adaptive loss function not only gives emphasis on the distance from the decision boundary of a sample's embedding but also the quality of the sample which was proxied by the magnitude of the normalized feature norm. If the image quality is high, then the model will give

more emphasis for samples far from decision boundaries (hard to predict) to capture fine details and if image quality is low the model will give more emphasis for samples near to the decision boundary (easy to predict) to ignore unidentifiable images and avoid destroying trained weights.

Advantages of Adaloss over Arcloss

The Arcloss is designed in such a way that the model prioritizes the samples which are wrongly classified and are close to the boundary layer and at the same time de-prioritizes the samples that are wrongly classified and are far from the boundary layer. This works well for high quality datasets and for low-quality datasets, one can expect to give good results by giving negative margin and trying to prioritize the samples that are wrongly classified and near to the margin. But for datasets where both high quality and low-quality samples are both present, we can only give either positive margin or negative margin and prioritize only 1 strategy.

Adaloss solves this problem by bringing image quality into the picture. As the margin is adaptive to each sample based on the image quality, Adaloss will train the model in such a way that it prioritizes hard to predict samples for high quality images and easy to predict samples for low quality images. This way, the model can achieve two things.

- It learns fine-grained features from the harder to predict high quality images, and
- It ignores unidentifiable images (hard to predict low quality images) which can degrade the learned features.

Implementation of the code

For the implementation, we have considered using the same parameters as the paper used throughout the code. If and only if the performance is bad, we used another parameters.

MNIST Dataset:

For MNIST Dataset we though Resnet18 will be a over complex model. So, we have created our own custom CNN as the backbone model. Our Custom CNN consists of two convolutional layers (32 and 64 filters) with ReLU as the activation function, each followed by max pooling, and finally a fully connected layer producing a 128-dimensional feature embedding. Initially, when no batch normalization and dropout layers were used and the results after 10 epochs looked as below:

	Loss	Accuracy
Train	0.08	99.87
Test	0.33	98.71

We can clearly see that the model is overfitted. So, we then introduced Batch normalization, and the results after 20 epochs were as below:

	Loss	Accuracy
Train	0.234	99.49
Test	0.362	98.83

The results improved, but there still was a sign of overfitting. We then introduced dropout set at 0.25. The results after 25 epochs are as below:

	Loss	Accuracy
Train	0.269	99.26
Test	0.235	99.22

Here clearly, the accuracy improved and there is no sign of overfitting and the Adaloss function performed very well on the MNIST dataset paired with our custom CNN as the backbone.

CIFAR10 Dataset:

For CIFAR 10 Dataset, we have utilized the ResNet18 model as the backbone as it was done in the AdaFace paper. As the ResNet18 model is very complex to CIFAR10 model there was huge overfitting as below (10 epochs):

	Loss	Accuracy
Train	0.421	96.25
Test	5.235	83.71

Then, we have introduced on-the-fly data augmentations as done in the paper (cropping, rescaling and photometric jittering). The results after augmentation are as below (10 epochs):

	Loss	Accuracy
Train	2.269	94.52
Test	4.932	82.64

We can still notice the occurrence of overfitting. Finally, we have introduced Batch Normalization layer, introduced a fully connected layer to reduce the dimensions from 512 to 256 for the embeddings and also introduced dropout of 0.6. Below are the results after 5 epochs: (after 5 epochs the model is over fitting)

	Loss	Accuracy
Train	4.5895	82.53
Test	5.7319	80.37