ArcFaceLoss

ArcFace: Additive Angular Margin Loss for Deep Face Recognition

```
losses.ArcFaceLoss(num_classes, embedding_size, margin=28.6, scale=64,
**kwargs)
```

Equation:

$$L_3 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n}e^{s\cos\theta_j}}$$

Parameters:

- **margin**: The angular margin penalty in degrees. In the above equation, `m = radians(margin)`. The paper uses 0.5 radians, which is 28.6 degrees.
- **num_classes**: The number of classes in your training dataset.
- **embedding_size**: The size of the embeddings that you pass into the loss function. For example, if your batch size is 128 and your network outputs 512 dimensional embeddings, then set `embedding_size` to 512.
- **scale**: This is `s` in the above equation. The paper uses 64.

Ways to improve the model :

Change the way we train : Data Augmentation
There exists a lot of ways to improve the results of a neural network by changing the way we train it. In particular, Data Augmentation is a common practice to virtually increase the size of training dataset, and is also used as a regularization technique, making the model more robust to slight changes in the input data.
Data Augmentation is the process of randomly applying some operations (rotation, zoom, shift, flips,…) to the input data. By this mean, the model is never shown twice the exact same example and has to learn more general features about the classes he has to recognize.

Change the way we test : Test Time Augmentation
The purpose of Test Time Augmentation is to perform random modifications to the test images. Thus, instead of showing the regular, “clean” images, only once to the trained model, we will show it the augmented images several times. We will then average the predictions of each corresponding image and take that as our final guess.
What happens now if we apply Test Time Augmentation ? We will present 5 slightly modified versions of the same image and ask the network to predict the class of each of them.
The reason why it works is that, by averaging our predictions, on randomly modified images, we are also averaging the errors. The error can be big in a single vector, leading to a wrong answer, but when averaged, only the correct answer stands out.