Copernican loss: Learning a Discriminative Cosine Embedding

Dipan K. Pal and Marios Savvides
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
{dipanp,marios}@cmu.edu

Abstract
Euclidean embeddings are ubiquitous in machine learning applications in classification settings. However, typical testing schemes (such as in face recognition) utilize the cosine similarity for feature matching, which creates a disconnect between training and testing scenarios. In this paper, we present Copernican loss, which learns a *discriminative cosine* embedding that addresses this gap. Copernican loss discriminates between all samples from different classes within a batch of size $m$ with just $m$ gradient computations (as opposed to $m^2 - m$ in previous work). We demonstrate the efficacy of the proposed approach with extensive experiments on CIFAR 10 and 100. Further, while keeping away from performance boosting preprocessing steps such as face alignment in training and testing, we match high performance on large scale face recognition datasets namely LFW and IJB-A Janus.

Introduction
Recent developments in deep neural networks have addressed a wide array of components. There has been considerable effort in developing deeper structures (Simonyan and Zisserman 2014), Szegedy et al. 2015, He et al. 2016) and more effective non-linearities (Goodfellow et al. 2013, Nair and Hinton 2010, He et al. 2015). Apart from structural developments, there have been many efforts in combating over-fitting and obtaining better gradients (Ioffe and Szegedy 2015, Srivastava et al. 2014, Salimans and Kingma 2016). Although fewer in number, there have also been recent studies recognizing the importance of stronger loss functions such as (Hadsell, Chopra, and LeCun 2006, Schroff, Kalenichenko, and Philbin 2015, Wen et al. 2016, Tadmor et al. 2016). Indeed, a robust loss function which encourages highly discriminative feature learning is a direct way to provide the network with more informed gradients towards the ultimate supervised task. A fully connected layer coupled with the cross-entropy loss and the softmax layer, together which we call the Softmax loss in this paper, is arguably the most prevalent loss function in practice. The Softmax loss has proved to be very versatile in use, and is able to provide reasonably good gradients owing to the well-behaved cross-entropy loss. A few recent studies have attempted to modify the Softmax loss in order to increase discrimination in terms of larger angular margin (Liu et al. 2016), or normalize the features going into Softmax thereby solving a non-convex problem (Ranjan, Castillo, and Chellappa 2017).

A different thrust towards obtaining highly discriminative features involves minimizing alternate loss functions or augmenting the Softmax with supplementary losses. Contrastive loss (Hadsell, Chopra, and LeCun 2006) and the Triplet loss (Schroff, Kalenichenko, and Philbin 2015) replace the Softmax loss with losses which focus on learning a discriminative embedding while trying to minimize intra-class variation in the learnt features. This is done by carefully sampling training pairs or triplet sets which leads to expensive hard-sample mining in large-scale applications. Center loss (Wen et al. 2016) on the other hand, is an approach which ignores hard-sample mining while only trying to minimize intra-class variation along with the Softmax loss. Another approach which was proposed to work on random batches is the Multi-batch estimator (Tadmor et al. 2016). Multi-batch is an example of the metric learning approach which for a batch size of $m$, utilizes all $m^2 - m$ pairs for a better estimate of the gradient. All these works mentioned operate in the $l_2$ or Euclidean space. As we explain in a later section and demonstrate in our experiments, although $l_2$ embeddings perform well in many applications, the performance gain they provide is limited in situations when the number of samples per class is high. In such situations, forcing all samples from a class to converge towards each other in the $l_2$ sense is too difficult a task since it requires the network having to converge not only in angle but also the norm of the features. Further, during testing in typical supervised classification such as face recognition, the most common metric is the cosine distance which ignores the norm. This creates a disconnect between training and testing since the network learns a behavior (to constrain the norm as well) that is ignored during test. Such a framework is inefficient. Recently, COCO (Liu, Li, and Wang 2017), a form of cosine loss was proposed for person recognition. COCO minimizes intra-class variation towards a class center and maximizes inter-class variation of samples with the centers of other classes as opposed to the global batch center which significantly raises computational complexity. The approach uses hard normalization and is similar.

---

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
to recent other studies (Ranjan, Castillo, and Chellappa 2017; Wang et al. 2017), all of which formulate a non-convex constraint.

**Contributions.** Our proposed Copernican loss has two important properties. 1) It is designed to augment the standard Softmax loss while *explicitly* minimizing intra-class variation and **simultaneously** maximizing inter-class variation. 2) It operates using the *cosine distance* and thereby directly affects the learning which removes the disconnect between training and testing. This improves efficiency since more of the model complexity is utilized to learn a more discriminative embedding rather than learning to constrain the norm. Copernican loss does not require hard sample mining or data augmentation of any kind, and can be effectively minimized using SGD on random mini-batches. It only needs to maintain a center for each class called the Planet center and computes the mean of the mini-batch called the Sun center (humoring the Copernican analogy of the solar system). In order to minimize *intra-class* variation, it minimizes the cosine distance of the samples to their corresponding Planet centers. In order to *discriminate* between the samples within a mini-batch, Copernican loss maximizes the cosine distance of the samples away from the mean of the mini-batch called (Sun center). This eliminates the need to compute a pair-wise gradients such as the Multi-batch (Tadmor et al. 2016) while providing the similarly discriminative gradients in a more efficient manner.

**Copernican loss: Learning a Discriminative Cosine Embedding**

**Motivation and Intuition**

**The Need for Simultaneous Discrimination and Invariance.** Learning robust features is a key problem in supervised learning. Robustness here refers to two specific properties of a useful feature. 1) *Invariance* to intra-class nuisance transformations while being 2) *discriminative* to inter-class transformations. Although there exist loss functions prevalent in practice *implicitly* optimize for this criteria, such as the Softmax loss and negative log likelihood, the features learnt using pure forms of the loss functions are not robust enough for harder classification tasks. Thereby, *explicit* simultaneous maximization of intra-class similarity and inter-class discrimination is critical. In contrast to some loss functions in literature (Liu et al. 2016; Wen et al. 2016), Copernican loss *explicitly* optimizes for both objectives (invariance and discrimination) between all samples in a mini-batch. Although, contrastive embedding (Hadsell, Chopra, and LeCun 2006) and Triplet loss (Szegedy et al. 2015) both advocate simultaneous optimization of discrimination and invariance, they search for useful gradients through clever and expensive sample mining. One way to mitigate the need for mining is to discriminate all samples belonging to different classes away from each other. Copernican loss does this by moving the samples away from the global batch mean, which efficiently provides discriminative gradients without sample mining.

**The Need for a Cosine Embedding.** Classification in machine learning, at its fundamental level, is typically conducted using the inner-product. The inner-product between the weight vector $w$ and the sample $x$ is a product of their norms and the cosine of the angle between the two vectors. The ideal classification vector $w$ would provide a high inner-product with a sample from the correct class and be low for ones from impostor classes. Thus we would want $w^T x_{cor} > w^T x_{imp} \Rightarrow ||x_{cor}|| \cos(\theta_{cor}) > ||x_{imp}|| \cos(\theta_{imp})$. The classification decision therefore, ultimately rests on the norm of the sample $x$ and the cosine angle between the weight vector. In this light, there are two ways of increasing discrimination. 1) Increase the norm of the correct class, and 2) decrease the cosine angle between the weight vector and the sample. For binary class problems, increasing the norm of one class over another might be feasible, however for multi-class problems this approach would not be effective. Maximizing the norm of samples from one particular class over all others, would hinder correct classification of other classes. Thereby, one approach to increase discrimination that can be applied to multi-class problems is to maximize the angle (or equivalently the cosine distance) between classes. This reasoning also applies to the Softmax loss function which is perhaps the most commonly used loss in supervised deep learning and is also the baseline in our study. Indeed, minimizing the intra-class cosine distance while simultaneously maximizing the inter-class cosine distance seems to be a reasonable goal. This is also exactly for our proposed Copernican loss optimizes for.

**Limitations on an $l_2$ embedding.** There have been multiple loss functions proposed that learn an $l_2$ embedding such as the Center loss (Wen et al. 2016). Triplet loss (Schroff, Kalenichenko, and Philbin 2015) and Multi-batch (Tadmor et al. 2016). All of these losses explicitly minimize the $l_2$ distance between samples from the same class. Concretely, for sample features $x_1$ and $x_2$ from the same class, $||x_1 - x_2||_2^2 = ||x_1||_2^2 + ||x_2||_2^2 - 2||x_1||_2||x_2||_2 \cos(\theta)$, where $\theta$ is the angle between the two samples. Minimizing this quantity requires 1) minimizing the difference between norm of the features $x_1$ and $x_2$ and 2) minimizing the cosine distance between the two. There are two cons of this approach.

1. During testing and extraction of a similarity score (*e.g.* for open set face or object feature extraction), only the cosine distance is taken into account. This implies that explicitly constraining the norm is inefficient for the loss function layer from the perspective of matching since the model needs to learn sub-tasks (*i.e.* constraining the norm) that it does not require during testing. Successful approaches such as batch normalization (Ioffe and Szegedy 2015) on the other hand do not require the *model or the weights* themselves to perform normalization, they perform it explicitly through the normalization operation. This allows the model complexity to be used to focus on the angles between the samples instead.

2. More importantly, for tasks with a large number of samples per class (*such as typical object recognition*), trying to

\[\text{In this paper, we jointly refer to the last fully connected layer of a deep network, along with the cross-entropy loss followed by a softmax layer as the Softmax loss.}\]
have the large set of samples per class all converge to the same point in the euclidean sense proves to be a difficult challenge for a deep network. As we find in our experiments with Center loss (Wen et al. 2016) it is much easier to simply have the angle of the sample features converge as opposed to convergence in the $l_2$ sense. With this approach, the network does not need to constrain norms, and model complexity is better utilized in creating a larger angular margin resulting in better performance during testing. Perhaps this is the reason that none of the previous works on learning $l_2$ embeddings (Wen et al. 2016, Schroff, Kalenichenko, and Philbin 2015, Tadmor et al. 2016, Hadsell, Chopra, and LeCun 2006) report results on object recognition datasets with a large number of samples per class. Examples include datasets such as CIFAR10 and CIFAR100 which have fewer classes (10 and 100 respectively) but many more samples per class (6000 and 600 images per class respectively). The focus of those studies is mainly face recognition which is characterized by a large number of classes (e.g. above 10,000 for CASIA-WebFace) with relatively few samples per class (average of about 50 samples per class for CASIA-WebFace).

Copernican loss

For a batch of size $m$, Copernican loss, denoted by $L_C$, is defined as the sum of three loses as follows.

$$L_C = L_{Soft} + \lambda \left( \frac{1}{m} \sum_{i} (1 - \cos(x_i, p_{y_i})) \right) + \left( \frac{1}{m} \sum_{i} \max(0, \cos(x_i, s) - \beta) \right)$$

Here, $L_P$ is the Planet loss minimizing intra-class cosine variation and $L_S$ is the Sun loss, maximizing inter-class cosine variation and $L_{Soft}$ is the Softmax loss. $\beta$ is the margin for Sun loss, $s$ is the global center for the particular batch (or the ‘sun’) and $p_{s}$ are the class centers (or the ‘planets’). In the ideal case, $s$ and $p_{s}$ would represent the class centers and the global center of the entire dataset. However, computing these quantities over the entire dataset would be very expensive, especially for large-scale applications. To get around this problem, in the case of the global center (sun $s$) of the entire dataset, we approximate it with the global center of each

---

In our experiments with Center loss, which was shown to perform well in face recognition tasks (Wen et al. 2016), we found that it consistently performs worse than Copernican loss (and for $\lambda = 0.1$ worse than vanilla Softmax loss) on both CIFAR10 and CIFAR100 a fact which supports our hypothesis.

---

4To humor our Copernican analogy, the samples themselves could be considered to be the ‘moons’ of their corresponding planets (or class centers).
Therefore for every batch, $s = \frac{1}{m}\sum_{i=1}^{m} x_i$. Computation of the class centers (planets $p$) also face a similar issue due to scale. However, class centers cannot be effectively estimated using a single batch due to high variance, especially in the early stages of training. One way to obtain a better estimate is to maintain a class center and update it with every batch in the following sense,

$$p_{y_i}^{t+1} = p_{y_i}^t + \frac{1}{n} x_i$$

Here, we assume there are $n$ instances of class $i$ in the minibatch. $\alpha$ is the update weight factor and is usually set to a small value (say 0.05). Compared to direct computation of class centers over the mini-batch set, this update provides more robustness to sample perturbation while simultaneously addressing the problem of scalability in estimating the centers. Copernican loss is simple to implement and unlike Triplet loss (Schroff, Kalenichenko, and Philbin 2015), it does not require hard sampling mining which would have increased the computation complexity of the overall loss. Further, computation of discriminative gradients only requires $m$ gradient computations compared to $m(m-1)$ for Multi-batch (Tadmor et al. 2016).

**Resolving an ambiguous configuration.** We discuss the ambiguous configuration presented in (Tadmor et al. 2016) in Fig. 1(b). Here, four samples are mapped into the feature space during early training when classes are mixed. The algorithm needs to pick pairs of samples from different classes to take a gradient step towards better discrimination. Picking Alice-Carol pair would not help the configuration as they have sufficient separation ($>1$) and neither will Bob-Alice pairs. As (Tadmor et al. 2016) points out, only Carol-Bob pairs will provide useful updates by moving Bob away. Center loss (Wen et al. 2016) would worsen things since it will push the two Alice’s closer to Bob. Triplet loss (Schroff, Kalenichenko, and Philbin 2015) would need to sample Carol-Bob which is expensive to determine hard-sample mining and unlikely if no sample mining is performed. Multi-batch (Tadmor et al. 2016) solves this problem by considering all pairs in the mini-batch. Our proposed Copernican loss is guaranteed to provide discriminative gradients for all samples without the need for 1) sample mining nor 2) $m(m-2)$ gradient computations (we need only $m$), since it moves samples away from the Sun center (black dot in Fig. 1(b)). To bring Alice pair closer, Planet loss provides a gradient which will be effective since Bob has moved out of the way due to the Sun loss.

**Optimization**

The gradients of Copernican loss are straightforward. Since the sun center and planet centers are updated during forward pass, only gradients with respect to the input $x_i$ is required to be derived which are as follows.

$$\frac{\partial L_C}{\partial x_i} = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{||x_i||_2} \left( \frac{p_{y_i}}{||p_{y_i}||_2} - \cos(x_i, p_{y_i}) \frac{x_i}{||x_i||_2} \right)$$

(3)

$$\frac{\partial L_S}{\partial x_i} = \left\{ \begin{array}{ll} \frac{1}{m} \sum_{i=1}^{m} \frac{1}{||x_i||_2} \left( \frac{s}{||s||_2} - \cos(x_i, s) \frac{x_i}{||x_i||_2} \right) & \text{if } \beta \leq \cos(x_i, p_{y_i}) \\ 0 & \text{otherwise} \end{array} \right.$$  

(4)

where $\frac{\partial L_C}{\partial x_i}$ becomes 0 only if $\cos(x_i, s) \leq \beta$. It is interesting to note that the gradient direction of the cosine similarity is different from the MSE gradient by a scaled version of the sample feature. This scaling factor is the cosine angle between the sample feature and the target vector. Fig. 2 shows the norm of the difference between the MSE gradient and the cosine similarity gradient as dimensionality increases along with the angle between the two.

For all dimensions above 10, the angle between the cosine similarity gradient and MSE gradient is about 20°. It is interesting to note that a gradient direction differing by 20° can result in significantly improved performance as we find in our experiments. As the sample features converged to the planets or class centers, the MSE gradient direction converges to the cosine similarity gradient direction, since $\cos(x_i, p_{y_i}) \rightarrow 1$. This fact coupled with superior performance of the cosine similarity leads us to hypothesize that during the initial stages of training, when the features are not informative of the class structure, MSE or $l_2$ embedding gradients are more noisy. Cosine embeddings are able to obtain better gradients by ignoring this noise since it only looks at angular margins which is not affected by the norm of the vectors. We observe this in our experiments.

**Experimental Validation**

We present results on general object recognition benchmarks of CIFAR10 and CIFAR100 primarily to study and compare the behavior of the Copernican loss with vanilla Softmax and Center loss (Wen et al. 2016). We also showcase the efficacy of our method on large scale face recognition tasks with no alignment for a harder challenge. We implemented Copernican loss and Center loss following (Wen et al. 2016) in Torch. The class center update learning rate $\alpha$ was set to 0.05 for all experiments.

**General model and implementation details.** Our benchmarking model for all experiments for consistency is a Resnet following (He et al. 2016) based off the Torch implementa-
During development of our method, similar performance trends were observed for other architectures as well. For CIFAR experiments, the models followed the corresponding CIFAR architecture from ResNet Torch codebase.

For our face recognition experiments our network had 4 stages with 2 blocks each, with number of filters as \{64, 128, 256, 512\}. The network ended with a global pooling stage (8 by 8 for CIFAR and 7 by 7 for face recognition) followed by a fully connected layer. The feature dimension (input to fully connected layer) for CIFAR was 1024 and for face recognition was 512. The bottleneck layer type is a pure ResNet connection. The CIFAR models were trained on a single GPU whereas the face recognition models were trained on 2 GPUs, all with a mini-batch size of 128, a weight decay of 0.0005 and momentum of 0.9. Learning rate was set to 0.1 and reduced by a factor of 10 at the 50% and at the 75% mark.

**Experiments with CIFAR 10 and CIFAR 100**

Our experiments with CIFAR10 and 100 focus on exploring the effect of \(\beta\) and \(\lambda\) on performance. We also focus on comparisons with the baselines Softmax and Center loss. A comparison with a version of Copernican loss learning an \(l_2\) embedding (instead of cosine) is also made. We train on default 50K training set and test on the 10K test set. The only data augmentation used was zero-padding the original image \(32 \times 32\) image to \(40 \times 40\) followed by a random crop of size \(32 \times 32\) and flipping with a probability of 0.5. Fig. 3 and Table 1 present the results of these experiments. The final error rate reported is the average of the last 50 epochs (with very small variance).

**Limitations of an \(l_2\) embedding.** To highlight the significance of a cosine embedding, we also benchmark against the \(l_2\) version of Copernican loss (denoted as \(l_2\) Copernican (Ours) in Table 1). The \(l_2\) version minimizes the MSE instead of the cosine distance between the sample features \(x_i\) and the planet centers \(p_k\), and maximizes the same between the features \(x_i\) and the sun center \(s\). From Table 1 we find that both Center loss and \(l_2\) Copernican loss perform worse than the cosine embedding. Interestingly, Table 1 shows that Center loss performs worse than vanilla Softmax for \(\lambda = 0.1\). This provides more evidence towards the limitations of an \(l_2\) embedding. Indeed, trying to converge all 6000 samples per class for CIFAR 10 and 600 samples for CIFAR 100 to the same point in euclidean space (the class center) in the \(l_2\) sense proves to be a difficult task which hampers overall network performance. A cosine embedding consistently provides significant improvement with the top performance being 5.04% on CIFAR 10 for \(\lambda = 1, \beta = -1\) and 23.68% on CIFAR 100 for \(\lambda = 5, \beta = -0.5\), thus showcasing its advantages.

**Effect of margin \(\beta\).** We explore the effect of margin \(\beta\) on Copernican loss. We fix \(\lambda = 0.1\) for Copernican loss and Center loss. Fig. 3(a) and Table 1 showcase the results of these experiments. Note that a lower \(\beta\) tries to enforce more discrimination. We find that a margin \(\beta = -1\) and \(\beta = -0.5\) perform the best on CIFAR 10 and 100 respectively. As a general observation, we found that if the number of classes are high, a lower margin performs better.

**Effect of loss weight \(\lambda\).** We explore the effect of \(\lambda\) on Copernican loss with \(\beta = -0.1\). Fig. 3(b) and Table 1 showcase the results of these experiments. For CIFAR, we find that a higher \(\lambda \geq 1\) improves performance for higher margin \(\beta\). Setting \(\lambda\) to be high for very high \(\beta\) led to a decrease in performance. Nonetheless, in all experiments with \(\lambda\) with \(\lambda\), the entire range significantly outperformed Softmax loss. As a rule of thumb, keeping \(\lambda\) low (around 0.1) for large scale experiments (as we find in face recognition) with a low margin works well, for smaller scale experiments (such as CIFAR), a higher \(\lambda\) offers more performance.

**Experiments in Large-Scale Face Recognition**

We benchmark Copernican loss on two large scale face recognition datasets namely LFW (Huang et al. 2007) and IJB-A Janus (Klare et al. 2015). Face recognition is a challenging task in general. This is because most benchmarks have a large number of classes (above 10,000 classes), with each class having a few number of samples (average 50 samples per subject for CASIA-WebFace, compared to say 6000 for CIFAR10). Such a task requires losses which can focus on the interplay between the few samples from different classes and is a good benchmark for comparison.

**Towards Alignment-free Face Recognition.** Most systems in face recognition utilize a landmark scheme for alignment (Schröff, Kalenichenko, and Philbin 2015; Taigman et al. 2014; Tadmor et al. 2016; Wen et al. 2016). Indeed, facial alignment boosts the performance of a face recognition

---

**Table 1**: Top 1 Test Error Rate (T.E.R) % on CIFAR 10+ and CIFAR 100+ (+ denotes standard data augmentation as in (Huang et al. 2016; He et al. 2016)). Numbers in brackets denote either \((\lambda)\) or \((\lambda, \beta)\). \(l_2\)-Copernican denotes the version which uses the Euclidean distance (baseline) as opposed to Cosine similarity (proposed).

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR 10+</th>
<th>CIFAR 100+</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoftMax</td>
<td>6.42</td>
<td>24.56</td>
</tr>
<tr>
<td>Center loss (0.1)</td>
<td>6.67</td>
<td>24.89</td>
</tr>
<tr>
<td>Center loss (1)</td>
<td>5.92</td>
<td>24.14</td>
</tr>
<tr>
<td>(l_2) Copernican (Ours) (0.1)</td>
<td>5.67</td>
<td>24.75</td>
</tr>
<tr>
<td>(l_2) Copernican (Ours) (1)</td>
<td>5.74</td>
<td>24.55</td>
</tr>
<tr>
<td>Copernican (Ours) (1, 0.5)</td>
<td>5.62</td>
<td>24.14</td>
</tr>
<tr>
<td>Copernican (Ours) (1, -0.1)</td>
<td>5.38</td>
<td>24.56</td>
</tr>
<tr>
<td>Copernican (Ours) (1, -1)</td>
<td>5.04</td>
<td>23.77</td>
</tr>
<tr>
<td>Copernican (Ours) (5, -0.1)</td>
<td>5.16</td>
<td>23.68</td>
</tr>
<tr>
<td>Copernican (Ours) (5, -0.5)</td>
<td>5.18</td>
<td>22.81</td>
</tr>
</tbody>
</table>

---

[^1]: https://github.com/facebook/fb.resnet.torch
engine by a significant amount which has led to saturation of benchmarks such as LFW. Although alignment is useful in its own right and also for real-world application, moving towards alignment-free pipelines offers a huge room for improvement and development and demonstration of more powerful methods. With this thought in mind, we restrict ourselves to the extremely challenging protocol of being alignment-free and utilizing minimal pre-processing of training and testing data. This leads to faster convergence.

**Training and pre-processing.** As mentioned, we utilize no alignment in either training or testing and therefore do not require any landmarking. All images we train and test on are 250 \times 250 square crops centered on the face. For cropping testing data, we utilized the bounding box provided, computed the center of the box and cropped a 250 by 250 patch around it. We train on the CASIA-WebFace (Yi et al. 2014) database containing about 10,575 subjects and 494,414 images which were originally 250 \times 250. Therefore the images were not processed any further and were only horizontally flipped for data augmentation. Due to the absence of alignment, there were many images with considerable in-plane rotation and/or slight scale variations which made the task much harder than the traditional aligned face recognition pipeline. This helps to showcase the efficacy of Copernican loss. Lastly, 127 was subtracted from each pixel value and the result was then divided by 127. We train models purely on CASIA-WebFace and test them on LFW and IJB-A Janus without any modification or finetuning.

**Testing Benchmark: LFW.** The LFW (Huang et al. 2007) database contains about 13,000 images collected from the web of 1680 distinct subjects. The database contains considerable degradations such as illumination, color, and age along with significant pose variation. We follow the unrestricted outside training data protocol for LFW exactly.

**Testing Benchmark: IJB-A Janus.** IJB-A (Klare et al. 2015) is a new difficult and publicly available challenge. IJB-A consists of 500 subjects under extreme conditions regarding pose, expression and illuminations with a total of 25,813 images. The IJB-A evaluation protocol mainly consists of face verification (1:1) and face identification (1:N). The interesting thing about this dataset is that each subject is described by a template containing a set of images or frames extracted from videos. We focus on the 1:1 tem-
Table 2: LFW results. Numbers in brackets denote either $(\lambda)$ or $(\lambda, \beta)$

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Data</th>
<th>Alignment</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FaceNet (Schroff, Kalenichenko, and Philbin 2015)</td>
<td>200M private</td>
<td>Yes</td>
<td>99.65</td>
</tr>
<tr>
<td>Center loss (Wen et al. 2016)</td>
<td>WebFace+0.2M private</td>
<td>Yes</td>
<td>99.28</td>
</tr>
<tr>
<td>DeepFace (Taigman et al. 2014)</td>
<td>4.4M private</td>
<td>Yes</td>
<td>97.35</td>
</tr>
<tr>
<td>DeepID (Sun, Wang, and Tang 2014)</td>
<td>88K</td>
<td>Yes</td>
<td>97.45</td>
</tr>
<tr>
<td>Masi et. al. (Masi et al. 2016b)</td>
<td>WebFace+2.4M synth</td>
<td>Yes</td>
<td>98.06</td>
</tr>
<tr>
<td>Multi-batch (Tadmor et al. 2016)</td>
<td>2.6M</td>
<td>Yes</td>
<td>98.20</td>
</tr>
<tr>
<td>Wang et. al. (Wang, Otto, and Jain 2015)</td>
<td>WebFace</td>
<td>Yes</td>
<td>97.52</td>
</tr>
<tr>
<td>Yi et. al. (Yi et al. 2019)</td>
<td>WebFace</td>
<td>Yes</td>
<td>97.73</td>
</tr>
<tr>
<td>DCNN (Chen, Patel, and Chellappa 2016)</td>
<td>WebFace</td>
<td>Yes</td>
<td>97.45</td>
</tr>
<tr>
<td>Center loss (Wen et al. 2016) (0.1)</td>
<td>WebFace</td>
<td>No</td>
<td>98.13</td>
</tr>
<tr>
<td>Copernican loss (0.1, 0.5) (Ours)</td>
<td>WebFace</td>
<td>No</td>
<td>98.27</td>
</tr>
</tbody>
</table>

Table 3: IJB-A Janus results. Numbers in brackets denote either $(\lambda)$ or $(\lambda, \beta)$

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Data</th>
<th>Alignment</th>
<th>VR (%) @ 0.1 FAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang et. al. (Wang, Otto, and Jain 2015)</td>
<td>WebFace</td>
<td>Yes</td>
<td>90.5</td>
</tr>
<tr>
<td>DCNN (Chen, Patel, and Chellappa 2016)</td>
<td>WebFace</td>
<td>Yes</td>
<td>80.0</td>
</tr>
<tr>
<td>DCNN (Chen, Patel, and Chellappa 2016)</td>
<td>WebFace</td>
<td>Yes</td>
<td>88.3</td>
</tr>
<tr>
<td>DCNN (Chen, Patel, and Chellappa 2016)</td>
<td>WebFace</td>
<td>Yes</td>
<td>94.7</td>
</tr>
<tr>
<td>DCNN (Chen, Patel, and Chellappa 2016)</td>
<td>WebFace</td>
<td>Yes</td>
<td>96.1</td>
</tr>
<tr>
<td>DCNN (Chen, Patel, and Chellappa 2016)</td>
<td>WebFace</td>
<td>Yes</td>
<td>96.7</td>
</tr>
<tr>
<td>PAM (Masi et al. 2016a)</td>
<td>WebFace</td>
<td>Yes</td>
<td>93.0</td>
</tr>
<tr>
<td>Chen et. al. (Chen et al. 2015)</td>
<td>WebFace</td>
<td>Yes</td>
<td>96.8</td>
</tr>
<tr>
<td>Center loss (Wen et al. 2016) (0.1)</td>
<td>WebFace</td>
<td>No</td>
<td>90.0</td>
</tr>
<tr>
<td>Copernican loss (0.1, 0.5) (Ours)</td>
<td>WebFace</td>
<td>No</td>
<td>95.0</td>
</tr>
</tbody>
</table>

References


