Learning a Metric Embedding for Face Recognition using the Multibatch Method

Oren Tadmor **Yonatan Wexler** **Tal Rosenwein**

Shai Shalev-Shwartz **Amnon Shashua**

OrCam Vision Technologies
13 Hartom St. Jerusalem, Israel

firstname.lastname@orcam.com

Abstract

This work is motivated by the engineering task of achieving a near state-of-the-art face recognition on a minimal computing budget running on an embedded system. Our main technical contribution centers around a novel training method, called Multibatch, for similarity learning, i.e., for the task of generating an invariant "face signature" through training pairs of "same" and "not-same" face images. The Multibatch method first generates signatures for a mini-batch of k face images and then constructs an unbiased estimate of the full gradient by relying on all $k^2 - k$ pairs from the mini-batch. We prove that the variance of the Multibatch estimator is bounded by $O(1/k^2)$, under some mild conditions. In contrast, the standard gradient estimator that relies on random $k/2$ pairs has a variance of order $1/k$. The smaller variance of the Multibatch estimator significantly speeds up the convergence rate of stochastic gradient descent. Using the Multibatch method we train a deep convolutional neural network that achieves an accuracy of 98.2% on the LFW benchmark, while its prediction runtime takes only 30msec on a single ARM Cortex A9 core. Furthermore, the entire training process took only 12 hours on a single Titan X GPU.

1 Introduction

Face representation through a deep network embedding is considered the state of the art method for face verification, face clustering, and recognition. The deep network is responsible for mapping the raw image of a detected face, typically after an alignment phase for pose correction, into a "signature" vector such that signatures of two pictures of the same person have a small distance while signatures of images of different individuals have a large distance.

The various approaches for deep network embeddings differ along three major attributes: first, and foremost, by the way the loss function is structured. Typically, the loss can be represented by the expectation of all pairs of "same" and "not same" images, or more recently as proposed by [11] as an expectation over triplets of images consisting of two "same" and a "not same" image. The second attribute is the network architecture: whether or not it assumes an alignment pre-process, whether it has locally connected layers [15], whether it is structured as a conventional concatenation of convolution and pooling [10, 17] or subsumption architecture [11]. The third attribute has to do with the use of a classification layer [15] trained over a set of known face identities. The signature

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

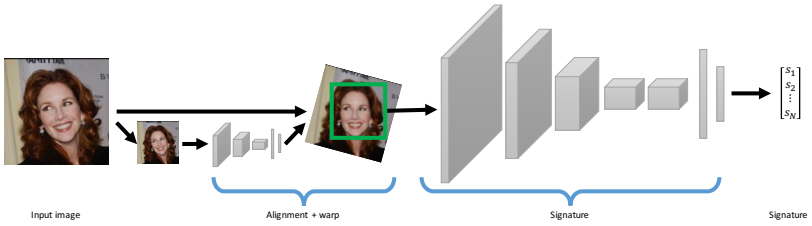

Figure 1: Overview of joint alignment and signature generation network

vector is then taken from an intermediate layer of the network and used to generalize recognition beyond the set of identities used in training.

Our goal in this work is to design a deep network embedding for face representation that is highly optimized in run-time performance, has a very high precision — close to the best, yet very expensive, networks and can be trained overnight using reasonable computing resources while using a training set of a manageable size. Specifically, using a training set of 2.6M face 112×112 images, running overnight on a single Titan X GPU card, our network has achieved an accuracy of 98.2% on the LFW benchmark (significantly higher than the best 2014 performance of [15] at 1/5000 of the running time) and it runs end-to-end using $41M$ FLOPs. This amounts to a runtime of 30ms on a single ARM Cortex A9 core (i.mx6 solo-light microprocessor by NXP, running at 1 GHz).

We train our network using the objective that requires the signatures of all "same" pairs to be below a global threshold while all the signatures of "not same" pairs should be above the global threshold. In Section 2 we show that fulfilling this objective is *harder* than training a multiclass categorization network over a set of known identities (the latter being the method used by [15] as a surrogate objective). While solving a multiclass categorization is an "easier" task, it requires a larger training set because of the significant increase in the number of parameters due to the large output layer (consisting of thousands of nodes, one per face identity).

Our main technical contribution centers around a better mechanism for constructing gradient estimates for the SGD method. The estimator, called Multibatch, is described in Section 3. We prove that the variance of the Multibatch estimator decreases (under some mild conditions) as $O(1/k^2)$. In contrast, the variance of the standard gradient estimate is order of $1/k$. The smaller variance of the Multibatch estimator, which comes with nearly no extra computational cost per iteration, is translated to a faster convergence of SGD, both in theory and in practice. This dramatic speed-up enables us solve the hard problem of metric embedding without relying on surrogates such as the multiclass surrogate of [15] or the triplet surrogate of [11].

From the engineering standpoint, we propose a streamlined network which combines the pre-processing step of face alignment into one network that can be optimized end-to-end. Specifically, the alignment pre-process is necessary for reducing the variability of the problem and thereby both reducing the size of the required training set and increasing the accuracy of the classifier. For the sake of unified implementation and optimization of code on the embedded platform we trained a small convolutional network with $4.8M$ FLOPs to produce the warping parameters as given by the state-of-the-art face alignment algorithm of [5]. The result is a system comprised solely of convolutional and linear transformations using a pipeline starting from a pre-process deep network, a warping transformation followed by the deep network embedding responsible for mapping the warped face image to a signature vector. The entire pipeline takes $41M$ FLOPs and runs on 112×112 face images with a runtime of 30ms on a single Cortex A9 core running at $1GHz$.

2 Learning a Metric

We seek to learn a transformation f_w, parameterized by w, from input space into $\mathbb{R}^d$. The resulting vectors should agree with "same-not-same" criterion where two samples from the same class should be close up to some learned threshold θ and vectors from differing classes should be farther than θ

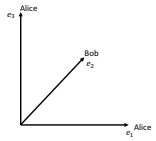

Figure 2: Why multiclass is easier than same-not-same: In this example, three samples of two identities are mapped to the canonical vectors e_i where Alice maps to e_1, e_3 and Bob maps to e_2. It is easy to see that $\|e_1 - e_2\| = \|e_1 - e_3\|$ so they are all equidistant. Still, a multiclass classifier is easy with $W_{\text{Alice}} = e1 + e3$ and $W_{\text{Bob}} = e2$.

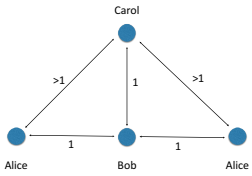

Figure 3: An ambiguous configuration. In this set, choosing either Alice-Bob pair will not improve the solution as both Alices will try to move closer to each-other while Bob will push them apart. Choosing Alice-Carol will not result an improvement since the distance is already larger than the margin. Only choosing Carol-Bob will improve the result. When choosing only pairs, the chance of choosing Carol-Bob is 1:3

under a norm of choice. Namely, for two samples x, x' and their labels $y, y' \in [k]$:

$$\begin{aligned} y = y' \quad &\Longrightarrow \quad \|f_w(x) - f_w(x')\|^2 < \theta - 1 \\ y \neq y' \quad &\Longrightarrow \quad \|f_w(x) - f_w(x')\|^2 > \theta + 1 \end{aligned} \tag{1}$$

The definition above satisfies our needs as it finds a metric embedding and a single global threshold. It enables any open-dictionary scenario, such as the case of face recognition. The existence of a global threshold is important as the typical scenario involves seeing many faces that are not known (e.g. people walking down the street).

We would like to optimize the loss over the parameters w of the network, including the value of the threshold θ. Given a training set $\{(x_i, y_i)\}_{i=1}^m$ we define the per-pair loss function:

$$l(w, \theta;\, x_i, x_j, y_{ij}) = \left(1 - y_{ij}\left(\theta - \|f_w(x_i) - f_w(x_j)\|^2\right)\right)_+ \tag{2}$$

where $y_{ij} \in \{\pm 1\}$ indicates whether the objects x_i, x_j are of the same class or not, and $(u)_+ := \max(u, 0)$. The global loss over the whole set is the average loss over all the pairs:

$$\mathrm{L}(w, \theta) = \frac{1}{m^2 - m} \sum_{i \neq j \in [m]} l(w, \theta;\, x_i, x_j, y_{ij})\ . \tag{3}$$

Learning a "same/not-same" classifier based on the above approach has been studied extensively. See for example [16, 13]. However, in the context of face recognition using deep networks, most previous works avoid using $L(w, \theta)$ directly and instead define an over-subscribed multiclass problem. For example [15] used a convolutional neural net to learn both w and a matrix-vector pair (W, b) such that $W \cdot f_w(x_i) + b$ is a vector whose largest coordinate is the correct class, $y_i \in [N]$, where N is the number of different labeled identities in the training set ($N = 4030$ was used in [15]). Then, a multiclass loss function is used, for example the multiclass hinge-loss:

$$\mathrm{MC}(w, W, b) = \sum_i \max_{t \in [N]} \left(1[t \neq y_i] + W_t \cdot f_w(x_i) + b_t - W_{y_i} \cdot f_w(x_i) - b_{y_i}\right)\ , \tag{4}$$

where W_t is the t'th row of the matrix W. Another popular multiclass loss is the logistic loss (which was used in [15]). The embedding $f_w(x)$ is the so-called last "representation layer", namely, the layer preceding the output layer. In [15] the representation layer has 4096 nodes. In other words, the embedding f_w is achieved indirectly through a multi-class problem with N different labels with the tradeoff of adding the matrix W with 4096×4030 parameters, which in fact is the majority of parameters of the entire network. The multiclass approach is therefore highly over-subscribed compared to the metric learning approach. We show next that indeed the metric learning approach is "harder" than the multiclass approach, explaining the motivation for learning embeddings in such an indirect manner.

Claim 1 *Metric learning is harder than multiclass in the following sense: Given w, θ for which Eq. 3 is zero, there exists (W, b) for which the value of Eq. 4 at w, W, b is zero. However, there exists w, W, b such that Eq. 4 is zero but such that for every θ, the value of Eq. 3 at w, θ is substantially greater than zero.*

Proof Let w, θ be such that Eq. 3 is zero. For every k, let C_k be the set of indices of examples that belong to class k. Define W to be the matrix such that its k'th row is $W_k = \frac{1}{|C_k|}\sum_{i\in C_k} f_w(x_i)$. Clearly, for every j

$$W_k \cdot f_w(x_j) = \frac{1}{|C_k|}\sum_{i\in C_k} f_w(x_i)^\top f_w(x_j)$$
$$= -\frac{1}{2|C_k|}\sum_{i\in C_k}\|f_w(x_i) - f_w(x_j)\|^2 + \frac{1}{2|C_k|}\sum_{i\in C_k}\|f_w(x_i)\|^2 + \frac{1}{2}\|f_w(x_j)\|^2$$

The last term in the above does not depend on k, and therefore it can be omitted since it does not effect the multiclass loss given in Eq. 4. Let $b_k = -\frac{1}{2|C_k|}\sum_{i\in C_k}\|f_w(x_i)\|^2$, and hence it cancels out the second term. It is left to deal with the first term. If $j \in C_k$ then $\|f_w(x_i) - f_w(x_j)\|^2 \leq \theta - 1$, hence the first term is at least $-\frac{\theta-1}{2}$. If $j \notin C_k$ then $\|f_w(x_i) - f_w(x_j)\|^2 \geq \theta + 1$, hence the first term is at most $-\frac{\theta+1}{2}$. Plugging into Eq. 4 we obtain that its value is zero. Finally, to show that the converse isn't true, consider Figure 2 which depicts three samples from two classes which are equidistant but are easily separable. ■

Our experiments show that optimizing Eq. 3 indeed converges slowly with SGD. While we later provide a mathematical analysis of this, let us first consider the typical and intuitive example in Figure 3. There, four points are mapped onto the target space, hence providing 6 pair-wise constraints. As is typical, the classes are mixed and a sample of Bob lies between two samples of Alice. It is easy to see that among all pairs, only the Carol-Bob pair is useful. Neither Alice-Bob pairs help as it would push Bob to be closer to another Alice (or have small effect pushing Bob outwards). An Alice-Carol pair would not result any loss as they are already far enough. Only by considering all pairs of samples, including Carol, that the correct update step can be reliably estimated.

The work of [11] used triplets in order to define an easier goal where pairs from each class should be closer than pairs from differing classes. That objective forgoes one global threshold and so is easier to solve. In order to achieve convergence they suggest a boosting scheme where "hard but not too hard" examples are picked using a separate process. In the next section we show how to extend this idea to an arbitrary number of pairs and solve the harder problem faster.

3 The Multi-Batch Estimator

In this section we describe the multi-batch approach for minimizing $L(w, \theta)$ (as given in Eq. (3)) and analyze its main properties. To simplify the presentation, denote $z = (w, \theta)$. Our goal is therefore to minimize $L(z)$. We also use the notation $\ell_{i,j}(z)$ to denote $\ell(w, \theta; x_i, x_j, y_{i,j})$.

The multi-batch method relies on the Stochastic Gradient Descent (SGD) framework, which is currently the most popular approach for training deep networks. SGD starts with some initial solution, $z^{(0)}$. At iteration t of the SGD algorithm, it finds an estimate of the gradient, $\nabla L(z^{(t-1)})$, and updates z based on the estimation. The simplest update rule is $z^{(t)} = z^{(t-1)} - \eta\hat{\nabla}^{(t)}$, where $\eta \in \mathbb{R}$ is a learning rate and $\hat{\nabla}^{(t)}$ is the estimate of $\nabla L(z^{(t-1)})$.

As indicated by the term "stochastic", the estimate of the gradient is a random vector, and we require that the estimate will be unbiased, namely that $\mathbb{E}[\hat{\nabla}^{(t)}] = \nabla L(z^{(t-1)})$. In words, on average, the update direction is the gradient direction.

We focus on a family of gradient estimates, in which the objective is being rewritten as an average of n functions, $L(z) = \frac{1}{n}\sum_{i=1}^{n} L_i(z)$, and the estimated gradient is obtained by sampling i uniformly at random from $[n]$ and setting $\hat{\nabla}^{(t)} := \nabla L_i(z)$. Due to the linearity of the derivation operator, we clearly have that this yields an unbiased estimate: $\mathbb{E}[\hat{\nabla}^{(t)}] = \mathbb{E}[\nabla L_i(z)] = \nabla\,\mathbb{E}[L_i(z)] = \nabla L(z)$.

In our metric learning problem, the objective, $L(z)$, is already defined as an average of $m^2 - m$ functions: for every pair $j, j' \in [m]$, $j \neq j'$, we can define $L_i(z) = \ell_{j,j'}(z)$. While this approach gives us an unbiased estimate of the gradient, the *variance* will be order of 1. In recent years, the importance of the variance of SGD algorithms has been extensively studied. Formally, the variance is defined as follows:

Definition 1 (variance of gradient estimate) *Let $L(z) = \frac{1}{n}\sum_{i=1}^{n} L_i(z)$. The variance of the natural unbiased gradient estimate is defined as:*

$$\nu^2(z) := \frac{1}{n}\sum_{i=1}^{n} \|\nabla L_i(z) - \nabla L(z)\|^2 .$$

Several recent works analyzed the convergence of SGD as a function of the variance. See for example [8, 14, 4, 12, 18, 9]. Ghadimi and Lan [1] have shown that even for non-convex problems, the number of iterations required by SGD to converge (in the sense of finding a point for which the squared norm of the gradient is at most ϵ) is order of $\frac{\bar{\nu}^2}{\epsilon^2} + \frac{1}{\epsilon}$, where $\bar{\nu}^2$ is an upper bound on $\nu^2(z^{(t)})$ for every t.

To decrease the variance, a common approach in deep learning is to rely on mini-batches. Formally, for any L of the form $L(z) = \frac{1}{n}\sum_{i=1}^{n} L_i(z)$, we can also rewrite L as $L(z) = \frac{1}{n^k}\sum_{i_1,\ldots,i_k} L_{i_1,\ldots,i_k}(z)$, where $L_{i_1,\ldots,i_k}(z) = \frac{1}{k}\sum_{r=1}^{k} L_{i_r}(z)$. This leads to the gradient estimate based on a mini-batch: sample $i_1, \ldots, i_k$ i.i.d. from $[n]$ and set the gradient to be the average value of $\nabla L_{i_j}(z)$, where averaging is over $j \in [k]$. Since the random vector is now an average of k i.i.d. random vectors, the variance decreases by a factor of $1/k$.

Getting back to our metric learning problem, estimating the gradient with a mini-batch of size $k/2$ requires the sampling of $k/2$ pairs of instances and calculating the loss for every pair. We observe that the main computation time for this operation is in the "forward" and "backward" calculations performed by the signature network over the k instances. The time requires to calculate the gradient of the loss given the signatures is negligible. This raises a natural question: if we have a budget of performing a forward and backward pass on k instances, can we find another unbiased estimate of the gradient whose variance is strictly smaller than order of $1/k$? The multibatch method aims at achieving this goal.

Intuitively, once we have calculated the signatures of k instances, it would be wasteful not to use the loss on all the $k^2 - k$ pairs of different instances. The multibatch estimate does exactly that — it samples k instances and defines the gradient to be the gradient of the average loss on all $k^2 - k$ pairs of different instances. Formally, let Π denote the set of permutations over $[m]$, and for every $\pi \in \Pi$ we define

$$L_\pi(z) = \frac{1}{k^2 - k}\sum_{i \neq j \in [k]} \ell_{\pi(i),\pi(j)}(z) .$$

The multibatch estimate samples π uniformly at random from Π and returns $\nabla L_\pi(z)$ as the gradient estimate. The following theorem shows that the multibatch estimate is an unbiased estimate and that the variance of the multibatch estimate can be order of $1/k^2$ under mild conditions.

Theorem 1

- $\mathbb{E}_\pi[\nabla L_\pi(z)] = \nabla L(z)$
- *For every i, r, denote $\bar{A}_i^{(r)} = \mathbb{E}_{j \neq i}(\nabla_r \ell_{i,j}(z) - \nabla_r L(z))$. Then,*

$$\mathop{\mathbb{E}}_{\pi} \|\nabla L_\pi(z) - \nabla L(z)\|^2 \leq \frac{1}{k^2 - k}\mathop{\mathbb{E}}_{i \neq j} \|\nabla \ell_{i,j}(z) - \nabla L(z)\|^2 + O\left(\frac{1}{k}\right)\sum_r \mathop{\mathbb{E}}_{i \sim [m]} \left(\bar{A}_i^{(r)}\right)^2 .$$

The proof of the theorem is given in Section A. The bound on the variance of the multibatch method is comprised of two terms. The first term is the variance of the vanilla estimate (based on a single pair) divided by $k^2 - k$. That is, for the first term we obtain a decrease of factor order of k^2. This is much better than the decrease of factor $k/2$ if we take $k/2$ pairs. It is easy to verify that the second term is at most $O(1/k)$ times the variance of the vanilla estimate. However, in most cases, we expect $\bar{A}_i^{(r)}$ to be close to zero. Intuitively, if for most of the i's, the expected value of $\nabla \ell_{i,j}(z)$ over $j \neq i$ is roughly the expected value of $\nabla \ell_{i,j}(z)$ over all pairs $i \neq j$, then the second term in the bound will be close to zero. We note that one can artificially construct z for which the variance is truly order of $1/k$, but such vectors correspond to bad metrics, and in practice, as long as the learning process succeeds, we observe a variance of order $1/k^2$.

4 Experiments on Face Recognition

Face recognition is an ideal test bench for similarity learning as it incorporates the need to map the input image into an invariant "signature" while meeting very high performance bars. Top scores on public benchmarks come mostly from large corporations with access to extensive resources both in the amount of training data and computing infrastructure. We demonstrate below that with the multibatch training algorithm we can accomplish near state-of-the-art precision using a fraction of the training set size, a fraction of the training time and a fraction of the prediction running time of top performing networks. We describe below the method for preparing a training set of $2.6M$ face images of $12k$ different identities, our method for incorporating face alignment into the process and the architecture of our convolutional network.

The leading approach for face recognition comprises of two steps. After detection, the first step is face alignment where the face is rectified to a fixed position and scale. This aligned image is then fed to the second step which computes a signature. Signatures are compared using a standard distance function in $\mathbb{R}^d$ to enable fast comparison in the open-dictionary. Figure 1 shows the network.

With regard to training sets, the available public datasets are not sufficiently substantial to allow high performance recognition. We follow the approach of [10] and search online for images of $12k$ public figures that are not included in the LFW benchmark [2]. As the search results are roughly sorted by decreasing reliability, we assume the first 50 of each identity are correct and train a binary SVM classifier to separate them from a random set of $10k$ other faces. This classifier is then used to pick correct images out of the average of 500 images per identity. We further leverage on [10] by using their published network as a feature and skip any manual filtering. This produced 2.6 million images. By manually verifying $4,000$ images we found 1.17% errors. For the sake of this work, we did not clean the set further.

With regard to alignment, previous work (cf. [15]) underscores the importance of image rectification pre-processing before being fed into a convolutional network. In [15] 3D rectification was proposed but more modest 2D rectification has been used by other systems as well ([10, 11]). Rectification allows the model to focus on inter-person difference rather than alignment hence relieving the classifier from requiring an ensemble of networks which are expensive at runtime.

We achieve the rectification goal through a 2D similarity transformation using a compact convolutional network with $102K$ parameteres taking $4.8M$ FLOPs which takes around 5msec run-time on a single Cortex A9 core. The network is trained on images of size 50×50 with desired output determined by the algorithm of [5] which uses random forests to detect the location of 68 facial landmark points. These points were used to compute warping parameters (rotation+translation+scale) which are then fed as ground-truth to train the ConvNet using a warping layer per [3]. The trained alignment network was used as the initial guess for the complete network described in Table 2.

While the code and model for [5] are available online, there are two advantages in replacing it with a ConvNet. First, from an implementation perspective it is natural to spend resources in optimizing a ConvNet on an embedded system rather than optimizing multiple algorithms. Second, the amount of run-time memory required by random forests is quite high for embedded use. It stands on $96MB$ in the available implementation — which is roughly a factor of 1400 compared to our ConvNet memory requirements.

Next, with regard to ConvNet architecture, the input layer consists of 112×112 RGB image and with an output layer of 128 nodes representing the face "signature". For the choice of network topology we follow the network-in-network ("NIN") concept of [6] which was later refined in [17]. The basic structure of a NIN consists of spatial convolutions followed by the same number of 1×1 convolutions. A typical NIN block is as follows: C convolution kernels with 3×3 spatial resolution which are typically strided followed by ReLU. The block is then passed through C 1×1 convolutions and ReLU. Table 2 presents the ConvNet architecture consisting of $1.3M$ parameters while taking $41M$ FLOPs which is roughly 30msec runtime on a single Cortex A9 core.

Training was done with multibatch, while setting $k = 256$. In choosing the 256 examples of the mini-batch, we randomly picked 16 individuals, and randomly picked 16 images per individual. We applied Eq. 2 as the loss per-pair, while weighting down the loss of "not-same" pairs so that FN and FP have the same impact, since equal error rate is the standard performance measure on LFW. We used learning rate of 0.01 throughout, and reduced it to 0.001 for the last epoch. For SGD

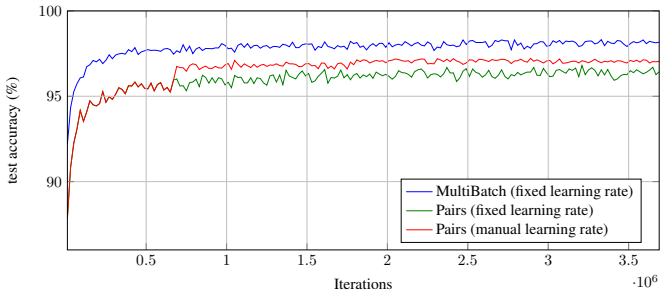

Figure 4: Multibatch convergence compared to using pairs. Here we show LWF Accuracy as defined in [2] by a 10-fold cross validation. The speed advantage of multibatch is clear, even when we manually reduced the learning rate of SGD tenfold whenever it stopped (at $0.7m$ and $1.7m$ iterations) to help it converge faster. This demonstrates how improved gradient estimation accuracy affects convergence.

Method	Training size	# Parameters	FLOPs	Ensemble	Total FLOPs	LFW Accuracy	Speed
MultiBatch	2.6M	**1.3M**	41M	1	**41M**	**98.20%**	1×
FaceNet [11]	260M	140M	1.6B	1	1.6B	99.63%	40×
DeepFace [15]	4.4M	120M	3.8B	5	19.3B	97.35%	4825×
VGG [10]	2.6M	133M	11.3B	30	340B	99.13%	8500×

Table 1: Comparison between leading published work on face recognition sorted by runtime.

we decreased the learning rate whenever it stopped decreasing in order to help it further. Figure 4 shows the test accuracy on the LFW dataset throughout the iterations of the multibatch training method compared to the standard pairwise SGD approach. One can clearly see that even with manual tweaking of the learning rate the pairwise SGD is significantly behind the multibatch algorithm. Both methods "flatten" around 1M iterations and further iterations do not help the pairwise SGD to recover and catchup with multibatch.

The ConvNet presented in Table 2 achieves a precision of 98.2% on the LFW benchmark with a runtime of 30ms on a single Cortex A9 ARM core. This result is superior to the 2014 state of the art achieved by [15] and better than human performance on this dataset. State of the art today achieves 99% and above but requires network ensembles at runtime, much larger networks, considerably larger training sets (e.g., $260M$ images used by [11]), and elaborate data augmentations such as multiple crops and resolutions. We also trained a much larger, VGG-like, model with $11B$ FLOPs which takes

Input Size	Type	FLOPs	Parameters
50×50×3	4 Convolutions, 5×5	760K	304
25×25×4	12 Convolutions, 5×5	758K	2K
24×24×12	12 Convolutions, 5×5	3M	4K
12×12×12	12 Convolutions, 3×3	189K	2K
6×6×12	4 Convolutions, 3×3	16K	436
6×6×4	Affine+ReLU, 256	38K	38K
256	Affine+ReLU, 64	17K	17K
64	Affine+ReLU, 140	10K	10K
140	Affine+ReLU, 64	10K	10K
64	Affine+ReLU, 128	9K	9K
128	Affine+ReLU, 64	9K	9K
64	Affine+ReLU, 4	260	260
112×112×3	Warp	150k	0
112×112×3	NIN with 32 kernels of 5×5 (stride 2)	8M	30k
56×56×32	max-pool 2×2	0	0
28×28×32	NIN with 96 kernels of 3×3 (stride 2)	8M	121k
14×14×96	NIN with 128 kernels of 3×3 (stride 2)	9M	165k
7×7×128	NIN with 128 kernels of 3×3	9M	165k
7×7×128	NIN with 128 kernels of 3×3	4M	165k
7×7×128	128 Convolutions, 3×3 (stride 2)	8M	148k
4×4×128	Affine+ReLU 256	525k	525k
256	Affine+ReLU 256	66k	66k
256	Affine 128	33k	33k
128	Loss		
Total:		41M	1.3M

Table 2: Our ConvNet architecture combines alignment and signature generation. The version here achieves an accuracy of 98.2% on LFW and takes 30ms on a Cortex A9 core. We also trained a much larger VGG-like model with $11B$ FLOPs which takes $6.6sec$ runtime and achieves 98.8% — an accuracy which appears to be limited by the noise in our training data (see text).

$6.6sec$ runtime and achieves 98.8% — an accuracy which appears to be limited by the noise in our training data.

5 Summary

The main technical contribution of this work centers around extracting a much smaller variance of estimating the gradient of the loss function for SGD for embedded metric learning. We have shown that our multibatch method introduces a dramatic acceleration of the training time required to learn an effective embedding for the task of face recognition. Along the way we have also addressed the issue of why practitioners tend to devise a classification network for face recognition rather than training a metric embedding network by showing that metric embedding is a harder problem to train — thus underscoring the importance of introducing the multibatch method.

References

[1] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[2] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[3] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *NIPS*, pages 2017–2025, 2015.

[4] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[5] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[6] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[7] Jingtuo Liu, Yafeng Deng, Tao Bai, and Chang Huang. Targeting ultimate accuracy: Face recognition via deep embedding. *CoRR*, 2015.

[8] Eric Moulines and Francis R Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011.

[9] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in Neural Information Processing Systems*, pages 1017–1025, 2014.

[10] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2015.

[11] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[12] Shai Shalev-Shwartz. Sdca without duality, regularization, and individual convexity. *arXiv preprint arXiv:1602.01582*, 2016.

[13] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the twenty-first international conference on Machine learning*, page 94. ACM, 2004.

[14] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599, 2013.

[15] Yaniv Taigman, Ming Yang, MarcAurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[16] Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15:505–512, 2003.

[17] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *ECCV 2014*, 2014.

[18] Peilin Zhao and Tong Zhang. Accelerating minibatch stochastic gradient descent using stratified sampling. *arXiv preprint arXiv:1405.3080*, 2014.