Siamese Networks For Chromosome Classification

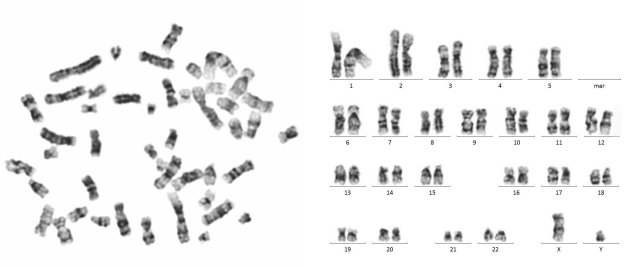
Swati, Gaurav Gupta, Mohit Yadav, Monika Sharma, Lovekesh Vig
TCS Research, New Delhi, India
Email : { j.swati | g.gupta7 | y.mohit | monika.sharma1 | lovekesh.vig }@tcs.com

Abstract

Karyotying is the process of pairing and ordering 23 pairs of human chromosomes from cell images on the basis of size, centromere position, and banding pattern. Karyotyping during metaphase is often used by clinical cytogeneticists to analyze human chromosomes for diagnostic purposes. It requires experience, domain expertise and considerable manual effort to efficiently perform karyotyping and diagnosis of various disorders. Therefore, automation or even partial automation is highly desirable to assist technicians and reduce the cognitive load necessary for karyotyping. With these motivations, in this paper, we attempt to develop methods for chromosome classification by borrowing the latest ideas from deep learning. More specifically, we perform straightening on chromosomes and feed them into Siamese Networks to push the embeddings of samples coming from similar labels closer. Further, we propose to perform balanced sampling from the pairwise dataset while selecting dissimilar training pairs for Siamese Networks, and an MLP based prediction on top of the embeddings obtained from the trained Siamese Networks. We perform our experiments on a real world dataset of healthy patients collected from a hospital and exhaustively compare the effect of different straightening techniques, by applying them to chromosome images prior to classification. Results demonstrate that the proposed methods speed up both training and prediction by 83 and 3 folds, respectively; while surpassing the performance of a very competitive baseline created utilizing deep convolutional neural networks.

1. Introduction

Conventionally, karyotyping of chromosomes is performed during the metaphase stage of cell division where the condensed chromosome images are Giemsa stained under a light microscope [24]. The Giemsa staining produces visible karyotypes where unique bands of dark and light color appear that help in distinguishing different chromosomes. A normal human cell has 46 chromosomes organized in 23 pairs of chromosomes, out of which 22 pairs are called autosomes and the 23^{rd} pair is the pair of sex chromosomes (X and Y). Karyotyping involves ordering of chromosomes in a standard format (i.e. enlisting into 24 categories) called a karyotype as shown in Figure 1.



(a) **(b)**

Figure 1. (a) Cell spread image during metaphase stage and (b) the standard and desired karyotyped image.

Doctors experienced in cytogenetics analyze microphotographed chromosome images to diagnose genetic disorders, specific birth defects, and cancers which arise due to various translocations, deletions or inversions [5]. Even after years of expertise, considerable manual effort and time is required to classify and order various types of chromosomes in order to produce a karyotyped image [24, 38]. With these motivations, this paper explores deep learning based methods for chromosome classification that can expedite the task of karyotyping, if not, automate it entirely. It is worth noting that such a system can seamlessly be employed to assist *expert doctors* and save their valuable time. In addition, research on karyotyping is also attractive because of its unique requirement of embodying human visual perception skills along with the domain expertise utilized by doctors [38]. As a result, karyotyping or chromosome classification is more specialized in comparison to any standard image classification task.

Although, deep learning methods have successfully managed to achieve state-of-the-art results for numerous computer vision tasks [31, 35, 19], it still continues to pose challenges in settings where data is limited [37]. Several tehniques have been proposed to adapt these models to low data regimes, under the umbrella of one-shot / few shot

learning [10, 32]. In particular, Siamese Networks [17] have made significant strides towards improved classification performance, on multiple benchmarks. Considering the empirical success of Siamese Networks and the practical settings where data for chromosome classification is scarce, we propose Siamese Networks to classify chromosomes.

More specifically, we submit that Siamese Networks preceded via a pre-processing step of straightening chromosomes [15, 16], is effective for the task of automatic chromosome classification. The rationale behind the pre-processing step of straightening of chromosomes, as opposed to an end-to-end approach is two fold: a) chromosomes obtained after Giemsa staining are found to be bent in different orientations which may deteriorate the performance of the classifier [2], b) it alleviates the requirement of learning representations invariant to different bending orientations. In particular, we deploy two straightening algorithms, namely, straightening via medial axis [15] followed by a manual rectification using the crowd for some selective cases that were difficult to automate, and straightening via projection vectors [16]. Subsequently, straightened chromosomes are fed into the Siamese Networks to perform automatic chromosome classification. During the course of above mentioned explorations, this paper makes the following contributions:

- We propose Siamese Networks for learning chromosome similarity for the task of automatic classification as described in Section 3.2.
- We employ a simple yet effective training method for Siamese Networks that advocates sampling from the dissimilar pairs in the Siamese training set to account for the skewed distribution of similar and dissimilar pairs for multi-class classification. The resulting technique yields an acceleration of 83 folds in training.
- A Multi-layer Perceptron (MLP) based feedforward network classifier is trained on the embeddings obtained from Siamese Networks, as opposed to a nearest neighbour search; thereby making the prediction step of Siamese Networks three times faster.
- We explore two straightening based pre-processing methods and comparatively benchmark Siamese Networks against Deep Convolutional Neural Networks (Deep CNN) [30] for the task of chromosome classification.

The remainder of the paper is organized as follows: Section 2 details an overview of related work for chromosome karyotyping. In Section 3, we present the proposed methodology, where we discuss straightening methods used for pre-processing and Siamese Networks for classification. Thereafter, in Section 4, we present details about the dataset, the training procedure adopted and a discussion on the obtained results. Finally, we place important conclusions and potential future research avenues in Section 5.

2. Related Work

Karyotyping of chromosomes during metaphase can be viewed as a task composed of two steps, namely, segmentation and classification of individual chromosomes. In the recent past, motivated from the fact that manual karyotyping is laborious and time-consuming; researchers have proposed computational methods for automatic karyotyping [4, 5] with results generally reported on very small datasets. While there exists numerous methods for automated segmentation [23] and classification of chromosomes [2, 25, 6], the unpredictable shapes and appearances of chromosomes due to their non-rigid nature, still continue to pose challenges for segmentation [4] as well as for classification [26, 1]. However, in this paper, we restrain our focus only on the step of chromosome classification for Karyotyping, with the assumption that we have been provided segmented images of individual chromosomes.

Research studies have revealed that the critical factor which impedes the performance of direct application of automatic chromosome classification methods, is the curved and bent orientations arising from the non-rigid nature of chromosomes [15, 16, 27]. Therefore, we have considered chromosome straightening as a pre-processing step for all the explored classification methods. [15] proposed a method to straighten chromosomes using their medial axis which is extracted by a thinning procedure operated on the binary version of the chromosome image. While the authors advocate its robustness for all types of chromosomes (bent, curved, highly-curved, multiple bends), this method fails if the underlying thinning procedure [14] yields a medial axis with spurious branches. We attempt to circumvent this issue by using a crowdsourcing platform to obtain corrections for chromosomes where we get medial axis with spurs / branches as a result of thinning. Authors in [16] and [27] have proposed another straightening method for chromosomes which utilizes vertical and horizontal projection vectors of binary images of the chromosomes obtained at various rotation angles, followed by stitching of the two arms and extrapolation of missing pixels using mean of neighbourhood pixel values.

While we utilize crowdsourcing for a small portion of our task of chromosome classification, crowdsourcing has been actively contributing in many ways to generate necessary annotations for various similar tasks such as image segmentation [29], mitosis detection [3] and nucleus detection [13]. For a more comprehensive view of crowdsourcing for computer vision, we refer our readers to [18, 33].

Though the earlier attempts of applying artificial neural networks for chromosome classification demonstrated their

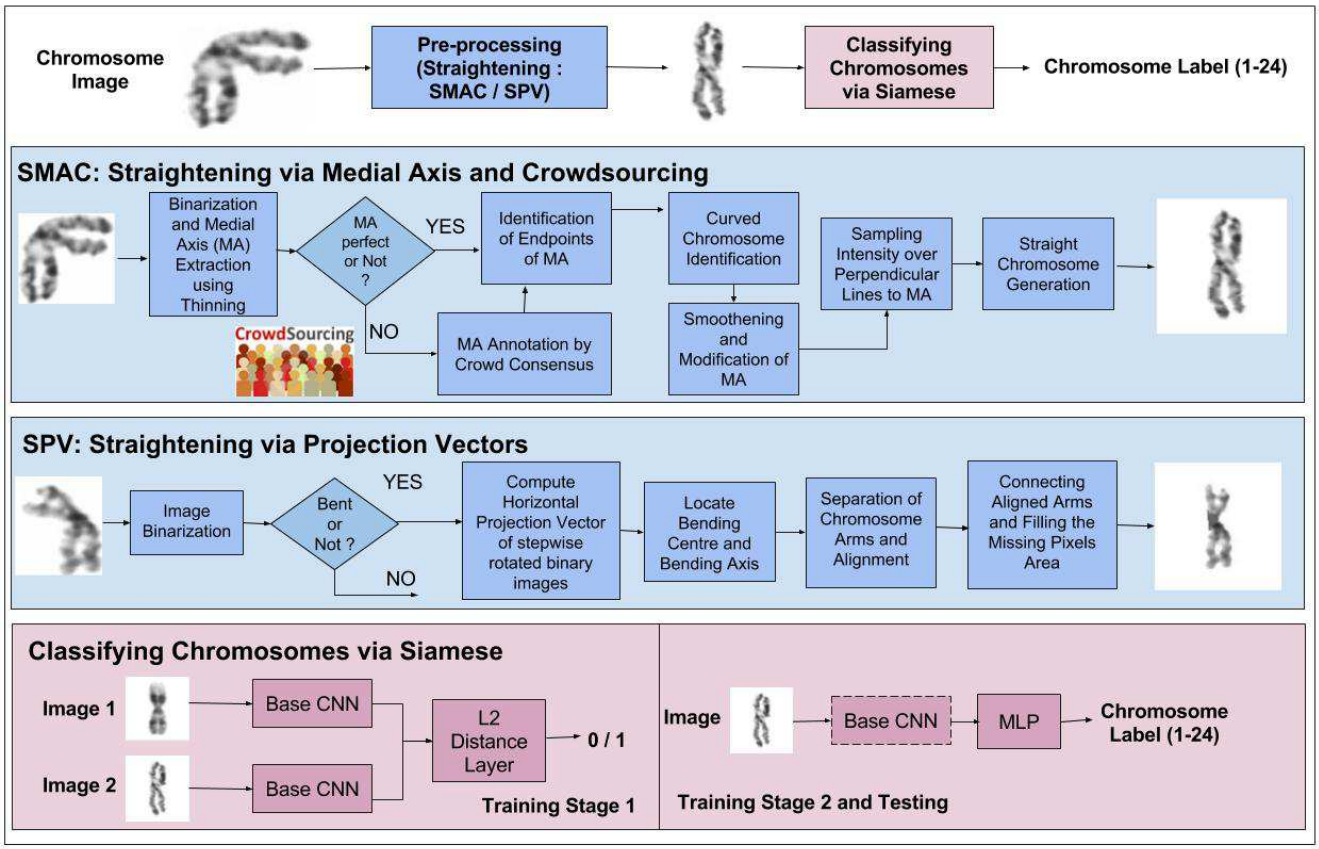


Figure 2. The flow-diagram of chromosome classification using Siamese-Networks: Chromosome images are pre-processed using two straightening algorithms - SMAC [15] followed by crowdsourcing for medial axis annotations and SPV [27]. The pre-processed images are then fed to Siamese Networks for classification. The architecture of the Base CNN used in Siamese-Networks is shown in Figure 5.

applicability [9], they were primarily based on manually designed features and feature selection methods [20]; similar to concurrent state-of-the art methods for image classification [21]. However, with the advent of deep learning techniques, Deep CNN [30] and similar architectures have surpassed the performance of conventional methods by a significant margin [19, 35, 31]. Further, these methods require a large amount of labeled data and fail to yield superior performance in the more practical settings where labeled data is scarce [10]. Therefore, researchers have proposed several architectures to handle limited data scenarios for image classification [17, 37, 28]. The arena of such one-shot or few shot learning methods is proliferating rapidly.

3. Proposed Method

This section provides details on the proposed method for the task of chromosome classification. The proposed method comprises of two stages: a) pre-processing images utilizing straightening methods and b) classifying chromosomes via Siamese Networks. In the first stage, we perform straightening on the available chromosome images utilizing the *Straightening via Medial Axis extraction and Crowdsourcing* [15] technique and the *Straightening via Projection Vectors* [27] technique, as explained below in Sections 3.1.1 and 3.1.2. Thereafter, processed chromosome images are fed into Siamese Networks to classify chromosomes as detailed in following Section 3.2. A detailed flow-diagram of all the steps in our proposed method is provided in Figure 2.

3.1. Pre-processing: Straightening Methodologies

3.1.1 *SMAC*: *S*traightening via *M*edial *A*xis extraction and *C*rowdsourcing

We have attemped to straighten chromosomes via Medial Axis (MA) extraction [15] which is applicable to all kinds of chromosomes. We have also considered manual corrections for MA extraction through crowdsourcing for some chromosomes where automated MA extraction failed. A flow diagram of the same is also outlined in Figure 2. Firstly, MA is automatically extracted using a standard thinning algorithm on the binarized chromosome images [14], which is succeeded by a Medial Axis smoothening

using a median filter of size 15×15. Considering MA is extracted using conventional thinning, some pixels are lost at both the ends of chromosomes, therefore smoothed MA is extended by 30 pixels on both the ends by calculating the slope of the MA using the last five pixels. Subsequently, the straightened chromosome is obtained by sampling the intensity of the original chromosome image over multiple closely located perpendicular lines on the modified MA. Next, pixels obtained after sampling are mapped into a matrix as rows to produce a straightened chromosome as shown in Figure 4 (a). However, such a method of straightening chromosomes is heavily dependent on MA extraction and sometimes the extracted MA has many spurious branches potentially due to some limitations of the thinning algorithm. For such cases, we employ the crowd for manual annotations of MA as described below:

Crowdsourcing for MA Extraction: We employ an internal crowd to manually extract / draw the MA line over chromosome images. Considering the lack of expertise of the crowd, it is likely that some workers may provide incorrect annotations. Hence, we used the Spammer Identification and Consensus [33] to validate the quality of obtained annotations. These measures, as discussed below, increase the cost marginally but ensure quality which is essential given the criticality of MA extraction.

- *Spammer Identification*: Spammers are those workers who provide completely spurious annotations. We have observed that annotations generated by spammers are minimally correlated with the chromosome region. Therefore, our filtering criteria for spammer identification is based on the intersection of chromosome-pixel locations (C_p) and annotation-pixel locations by workers (W_p). Specifically, a worker is identified as a spammer iff $||C_p \cap W_p||_0 \leq \alpha * ||W_p||_0$, where α is set to 0.6.

- *Consensus*: Every chromosome image is assigned to multiple workers to draw the MA of the chromosome. If there are K different workers then the most probable MA is the one which has the most proximity with the other MA drawn by $K-1$ workers. Assuming a medial axis X_i is drawn by i^{th} worker, we represent its normalized Hausdorff distance [12] from the medial axis drawn by the j^{th} worker as D_{ij}. We select consensus MA corresponding to the l^{th} worker which is determined according to the following Equation 1.

$$l = \arg\min_{i} \sum_{j=1, j\neq i}^{K} max\{D_{ij}, D_{ji}\} \qquad (1)$$

The Hausdorff distance D_{ij} is defined as the maximum of all the distances from a point in one set to the closest point in the other set. D_{ij} is directional as $D_{ij} \neq D_{ji}$, therefore we apply the max operation in Equation 1. A sample of MA annotations and the selection of final annotation using census is shown in Figure 3.

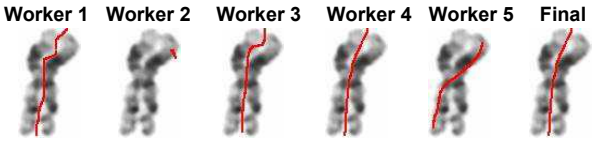


Figure 3. Sample MA annotations provided by the crowd for a chromosome image and the selected MA (*Final*) using consensus.

3.1.2 *SPV*: *S*traightening via *P*rojection *V*ectors

The SPV algorithm [27] begins with the binarization of a chromosome image and uses a technique to determine if the chromosome is bent or straight as shown in Figure 2. Since a tight bounding box of a straight chromosome contains less blank area as compared to the area for bent chromosomes in a binarized image, we have used a *Whiteness value* (V) [30] which is defined as the sum of white pixels divided by the total area of the tight bounding box of a binarized chromosome. We label the chromosomes with $V \leq V_T$ as bent chromosomes and they are sent for further straightening, where the value of whiteness threshold V_T is determined empirically.

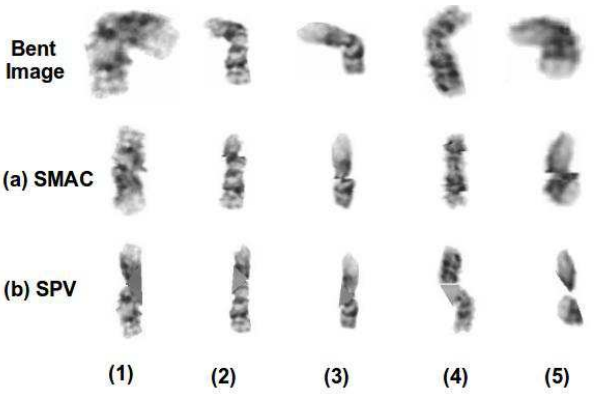


Figure 4. Examples showing results of straightening algorithms - (a) SMAC and (b) SPV.

After bent chromosomes are recognized, we compute the horizontal projection vectors of the step-wise rotated binary image and locate the bending centre and bending axis of the bent chromosome. The global minimum point in between two globally maxima points of the horizontal projection vector with their amplitudes almost equal corresponds to the bending centre of a curved chromosome. The horizontal line passing through the point on the chromosome image coincident with the global minimum point in the horizontal projection vector represents the bending axis of the

chromosome. The chromosome image is split into two sub-images along the bending axis, containing one arm each. Two sub-images are then rotated while calculating the vertical projection vector at each rotation step to align both the arms for stitching together. Due to the particular shape of each arm of the chromosome, the vertical projection vector will demonstrate minimum width if the arms are in the vertical position inside the sub-image. Subsequently, both the sub-images containing aligned arms are connected to produce the final straightened chromosome.

The resulting straightened chromosome contains some missing pixel values after the stitching operation which are filled with the mean value of neighbourhood pixels at the same horizontal level. Some examples of bent chromosomes straightened via SPV are shown in Figure 4(b).

3.2. Classifying Chromosomes via Siamese Networks

Siamese Networks are comprised of twin neural networks which learn to predict whether or not a pair of input images are similar or dissimilar. These twin neural networks are conjuncted through an energy function that is computed using a metric between the representations learnt at the highest layer. Since the model parameters for twin networks are tied, they are bound to learn the same transformation hence making it very likely that similar images will be mapped closer to each other in the learned feature space. For example, if a pair of chromosomes with very similar size, position of centromere, and banding patterns is fed to the Siamese Network then their feature representations are likely to be closer. Hence, energy at the top layer will be lower; as a result of which they will be potentially predicted as a similar pair. Another aspect of Siamese Networks is that it is symmetric with a top conjoining layer making it invariant to possible permutations between input images and twin networks.

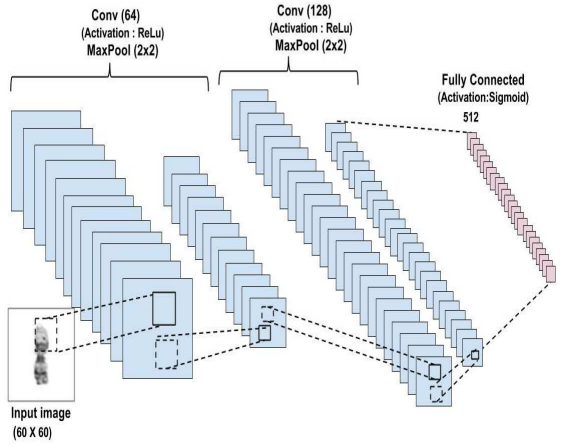


Figure 5. The *base CNN* architecture utilized in Siamese Network.

Our Siamese Network exploits a standard CNN as the base network. For each of the twin networks, let L be the number of layers with N_l units in layer l; $h_{1,l}$ and $h_{2,l}$ denote the hidden vector representations in layer l for the first and second twin, respectively. More specifically, the base CNN model utilized for twin networks has two convolutional layers consuming a single channel with filters of sizes constrained to multiples of 16. Each convolutional layer operates on a stride of 1 and is followed by a max pool layer of size 2×2, which is also operating at a stride of 1. All convolutional layers in our base CNN have Rectified Linear Units (ReLU). The output of the final max pool layer is flattened into a vector and fed into the succeeding fully connected dense layer having sigmoid units. The dense layer is followed by the computation of an energy function over the feature representations of highest level. A schematic diagram of the *base CNN* is provided in Figure 5.

To learn Siamese Networks, we optimize the contrastive loss function [8] with respect to the model parameters. The contrastive loss function ensures not only that the energy for a pair of inputs belonging to the same class should be low, but also that the energy for a pair from different classes is high. Unlike conventional machine learning algorithms where the loss function is calculated as the sum over samples, the loss function here is computed over pairs of samples as mentioned in Equation 2. Let $S = ((X_1, y_1)...(X_N, y_N))$ represent the training dataset, where X_i is an image with class y_i. Also, for a pair of images X_i and X_j, the binary label y_{ij} is assigned to 1 if both X_i and X_j are similar, i.e. if they belong to the same class, otherwise it is set to 0. Then, the contrastive loss function for our Siamese Network model can be defined as follows:

$$L(S, W) = \sum_{i=1}^{N} \sum_{j=i}^{N} E_W(y_{ij}, X_i, X_j) \tag{2}$$

where $E_W(y_{ij}, X_i, X_j)$ is an energy function between a pair of images X_i and X_j, defined based on euclidean distance; and W represents model parameters. More precisely, $E_W(y_{ij}, X_i, X_j)$ is calculated as in Equation 3.

$$\begin{aligned} E_W(y_{ij}, X_i, X_j) = \frac{1}{2} \times (1 - y_{ij}) \times (D_W(X_i, X_j))^2 \\ + \frac{1}{2} \times y_{ij} \times (\max\{m - D_W(X_i, X_j), 0\})^2 \end{aligned} \tag{3}$$

where m is the margin and $D_W(X_i, X_j)$ is the euclidean distance between the representations learnt by the base CNNs of the Siamese Network. $D_W(X_i, X_j)$ is computed as follows:

$$D_W(X_i, X_j) = \| h_{1,L}(W, X_i) - h_{2,L}(W, X_j) \|_2 \tag{4}$$

The model parameters W are shared by both of the twin networks in the Siamese Network, resulting in fewer parameters to learn. As a consequence, Siamese Networks require relatively less data to train and are less susceptible to overfitting. For prediction, typically a k-nearest neighbour (KNN) approach is utilized in the embeddings space learnt by the base CNN. Therefore, prediction time is large as it requires comparison with all the training samples, unlike a standard classification model such as an MLP classifier. Further, applying KNN requires training embeddings obtained from Siamese Network to be retained in memory, increasing the memory footprint of the model significantly. To circumvent these issues, we have applied a secondary training stage as depicted in Figure 2. During the second training stage, we train a two layer MLP based feedforward network classifier on top of the embeddings obtained from the base CNN. With this, the requirement to keep embeddings corresponding to training data is obviated, additionally the prediction step of an MLP is significantly faster.

Next, we observe that the contrastive loss function as mentioned in Equation 2, has a relatively higher number of dissimilar pairs in comparison to similar pairs; a scenario similar to skewed data distribution over classes [11]. To address this issue, we define a new modified loss function $\hat{L}(S, W)$, as mentioned in Equation 5:

$$\hat{L}(S, W) = \sum_{i=1}^{N} \sum_{j=i}^{N} E_W(y_{ij}, X_i, X_j) \times S(y_{ij}, X_i, X_j) \quad (5)$$

where $E_W(y_{ij}, X_i, X_j)$ is computed as mentioned in Equation 3. $S(y_{ij}, X_i, X_j)$ is an indicator function to decide whether or not a pair of sample images is to be included in the training set. Note that, such a function can be precomputed and the pairwise energy $E_W(y_{ij}, X_i, X_j)$ will be computed only for the cases that are part of the training, without increasing the computation overload significantly. For simplicity, we considered three scenarios. First is when $S(y_{ij}, X_i, X_j)$ is set to 1, which reduces to the loss function defined above in Equation 2. Second is where we randomly sample $S(y_{ij}, X_i, X_j)$ from a Bernoulli distribution, and reduce the ratio of dissimilar to similar pairs equal to R while including all the available similar pairs. If α and β are the number of similar and dissimilar pairs then the probability of $S(y_{ij}, X_i, X_j) = 1$ is defined as follows in Equation 6:

$$Pr(S(y_{ij}, X_i, X_j) = 1) = \begin{cases} 1 & \forall y_{ij} = 0 \\ \frac{R \times \alpha}{\beta} & \forall y_{ij} = 1 \end{cases} \quad (6)$$

Next in the third case, in addition to sampling $S(y_{ij}, X_i, X_j)$, we have taken the same number of dissimilar pairs of images from all the possible ${}^{24}C_2$ different class label pairs, thereby balancing the distribution not only between the similar and dissimilar pairs, but also the distribution of dissimilar pairs across ${}^{24}C_2$ different class label pairs. Though, we confine our exploration to make use of the second and third cases only once to select the training set, it will be trivial to extend their usage at the start of every epoch permitting the balanced training in addition to the usage of all the available possible pairs of images.

4. Experiments

This section starts with the description of the dataset collected from a hospital in subsection 4.1. Later in subsection 4.2, we explain the experimental setup utilized to conduct our experiments. Subsection 4.3 presents the obtained results while discussing the effectiveness of the proposed Siamese Networks for the task of chromosome classification. Next in Subsection 4.4, we visualize embeddings obtained from Siamese Networks, to ascertain their efficacy in achieving the separability necessary for the classification.

4.1. Dataset

To conduct our experimentation, we have collected chromosome images of real world healthy patients from a hospital. Total number of chromosome images are 1740 which are randomly divided into a group of 1296, 235 and 209; for training, validation and test sets, respectively. These images were segmented manually and a subset was validated with doctors to ensure that the label for each segmented individual chromosome image is identifiable. Each image is assigned a label from the 24 categories shown in Figure 1(b). For all the experiments, the resolution of chromosome images was set to 60×60 in grey-scale.

4.2. Training Details

All models were trained using stochastic gradient descent with a learning rate in $\{10^{-1}, 10^{-2}, 10^{-3}\}$, Nestrov momentum value set to 0.99, weight decay in $\{10^{-2}, 10^{-3}\}$. In addition, we apply dropout [34] to regularize the network training, and provide a sufficiently large number of epochs while training each model. Specifically, models in rows 1-3, row 4, rows 5-6, row 7, rows 8-11 and rows 12-15 of Table 1 were trained for 2000, 5000, 2000, 200, 25 and 100 epochs, respectively. We observe validation results at each epoch and track model parameters corresponding to highest validation accuracy, which were later used for testing. The margin value m of the Siamese Networks was set to 0.5. All methods are implemented using Theano [36] and Keras [7].

4.3. Results and Discussion

This section presents the obtained empirical results to validate the contributions mentioned above in Section 1. Table 1 shows percentage testing accuracy, training time (shown as *Epochs*×*PET*, where Epochs and PET refer to the number of epochs at which best model parameters are

Network Architecture	Similar:Dissimilar (1:R)	Accuracy	Epochs × PET	Prediction
1. Two Layer MLP	– – –	59.7%	1921×0.3 sec (0.16 hrs)	4.088 sec
2. Two Layer MLP + SMAC	– – –	67.9%	1871×0.3 sec (0.16 hrs)	4.088 sec
3. Two Layer MLP + SPV	– – –	72.3%	1791×0.3 sec (0.15 hrs)	4.088 sec
4. Deep CNN	– – –	68.5%	4700×4.1 sec (5.35 hrs)	12.085 sec
5. Deep CNN + SMAC	– – –	78.4%	1832×4.1 sec (2.09 hrs)	12.085 sec
6. Deep CNN + SPV	– – –	83.7%	1957×4.1 sec (2.23 hrs)	12.085 sec
7. Siamese Network + Nearest + SPV	36.6K:817K (Random)	**85.6%**	155×2844 sec (124.17 hrs)	15.088 sec
8. Siamese Network + MLP + SMAC (Avg)	36.6K:73.2K (Random)	78.5±0.8%	18.2×365 sec (1.84 hrs)	4.760 sec
9. Siamese Network + MLP + SMAC (Max)	36.6K:73.2K (Random)	79.4%	19×365 sec (1.92 hrs)	4.760 sec
10. Siamese Network + MLP + SPV (Avg)	36.6K:73.2K (Random)	81.3±0.7%	17.4×365 sec (1.76 hrs)	4.760 sec
11. Siamese Network + MLP + SPV (Max)	36.6K:73.2K (Random)	83.8%	18×365 sec (1.83 hrs)	4.760 sec
12. Siamese Network + MLP + SMAC (Avg)	9K:18K (Balanced)	78.6±0.9%	62.8×90 sec (1.57 hrs)	4.760 sec
13. Siamese Network + MLP + SMAC (Max)	9K:18K (Balanced)	80.4%	63×90 sec (1.58 hrs)	4.760 sec
14. Siamese Network + MLP + SPV (Avg)	9K:18K (Balanced)	**83.3±1.2%**	61.2×90 sec (1.53 hrs)	4.760 sec
15. Siamese Network + MLP + SPV (Max)	9K:18K (Balanced)	**85.2%**	**60×90 sec (1.50 hrs)**	**4.760 sec**
16. Con-Siamese Networks + MLP + SMAC	9K:18K (Balanced)	79.8%	–	–
17. Con-Siamese Networks + MLP + SPV	9K:18K (Balanced)	**84.6%**	–	–

Table 1. Results – Percentage test accuracy, training time (mentioned as Epochs×PET, where Epochs and PET represents number of epochs at which the best model parameters are obtained and per epoch time respectively), and prediction time (mentioned as Prediction), for the proposed Siamese Networks along with the multiple baselines created using a Deep CNN for the automatic chromosome classification.

obtained, and per epoch time, respectively), and prediction time for test samples (shown as *Prediction*), for the proposed Siamese Networks along with the multiple baselines created using *Two Layer MLP* and *Deep CNN*, for the task of chromosome classification. Our first set of baselines (rows 1-3) are from a *Two Layer MLP* based feedforward neural network classifier. In Table 1, rows 4-6 mention results of the *Deep CNN*, which comprises of 2 convolutional layers with 64 filters, a max pool layer, a dropout layer with 0.25 probability, 2 convolutional layers with 32 filters, max pool layer, dropout layer with 0.25 probability, 2 convolutional layers with 16 filters, max pool layer, dropout layer with 0.25 probability, 2 dense layers of size 1024 and 512, dropout layer with 0.5 probability, dense layer of size 24. All filters in convolutional and max pool layers are of size 3×3 and 2×2 respectively. The activation units for convolutional and dense layers have ReLU and sigmoid as non-linear transformations respectively. For both *Two Layer MLP* and *Deep CNN*, their *SMAC* and *SPV* variants outperform no straightening variants by a significant margin of absolute improvement ranging between 8.2% – 15.2%, making it evident that straightening is vital.

Next, row 7 of Table 1 reports the performance of the proposed *Siamese Network + Nearest + SPV* which is a Siamese Network applied on the chromosomes processed using the SPV method explained in Section 3.1.2. This method achieves an absolute and relative improvements of **1.9%** and **11.65%** respectively, when compared with the best baselines obtained utilizing a Deep CNN network. However, training and prediction timings of this network have increased to 124.17 hrs and 15.088 sec respectively, which are intractably high. Such a huge training time makes it unwieldy for practitioners to explore different choices of hyper-parameters. Moreover, its hefty prediction time is not viable enough for its deployment for a real-time use-case.

In Table 1, in rows 8-11, Siamese Networks are followed by an MLP based two layer feedforward networks, which can predict **3.16 folds** faster than the Siamese Network and nearest neighbour model. For all of these three cases, we have downsampled dissimilar image pairs by a factor of 11.62 (to reach 1:2 ratio of similar and dissimilar pairs, i.e. $R = 2$ which is mentioned as *Random* in Table 1). Hence, per epoch time is tremendously reduced and training time is made tractable. To account for the downsampling bias, we have trained 10 models on different sets of dissimilar image pairs. We report time for the average cases as *Siamese Network + MLP + SMAC (Avg)* & *Siamese Network + MLP + SPV (Avg)*, and best cases as *Siamese Network + MLP + SMAC (Max)* & *Siamese Network + MLP + SPV (Max)*, respectively. While these cases ameliorate the problem of intractable training and prediction time, their accuracy is lower than the *Siamese Network + Nearest + SPV*, which is when Siamese is trained on the all possible similar and dissimilar pairs. Still, it is noteworthy that even with a downsampling of 11.62 factor in dissimilar pairs of the training set, the best cases of the Siamese Network exceed the performance of the Deep CNN for the corresponding preprocessing methods (i.e. row11 > row6 and row9 > row5); indicating the superiority of the proposed Siamese Network based method for chromosome classification.

After that, we downsample the training set leaving a mere *24.5%* and *2.2%* of all the possible similar and dis-

similar pairs respectively, (mentioned as *Balanced*) in the 12-13 rows of Table 1. As a consequence of that, training time is reduced further. Also, the accuracy is improved potentially because of the performed balancing in the training set across different pairs of classes. Again, we report results of the average and best cases estimated using the performance obtained by training 10 models. One must notice that training time is improved to an extent where it is smaller than that of Deep CNN while maintaining the superior performance of Siamese Networks (e.g. see row 15 of Table 1, i.e. *Siamese Network + MLP + SPV (Max)*).

Following this, we conduct another experiment in which embeddings obtained from all the previously trained 10 Siamese Networks are concatenated and fed into a two layer MLP network, which is termed as *Con-Siamese Networks + MLP + SMAC* and *Con-Siamese Networks + MLP + SPV* for both *SMAC* and *SPV* methods, respectively. *Con-Siamese Networks + MLP + SPV* yields an accuracy of **84.6%** outperforming all the three strong baselines of Deep CNNs, a yet another validation showing that Siamese Networks have potential to outperform *Deep CNN* models.

4.4. Visualizing Siamese Embeddings

To probe the embedding representations learnt by Siamese Networks, we compute hidden representations corresponding to all the chromosomes present in the test set. We have utilized model parameters from the *Siamese Network + MLP + SPV (Max)* method mentioned in Table 1 and applied t-SNE [22] with default settings, to reduce the dimensionality of hidden representations to two dimensions. The visualization is depicted in Figure 6.

In Figure 6, it is interesting to observe that chromosomes of type 9 and 23 (denoted as $\triangledown$ and $\diamond$) as well as chromosomes of type 22 and 19 (denoted as $\star$ and $\blacksquare$), are closely embedded as they also have similar visual appearance, which can be observed in Figure 1 (b). Besides that chromosomes of type 1 and 19 (denoted as $\triangle$ and $\blacksquare$) are far apart in the obtained visualization as they are very different from each other. Such a behaviour of the learned embedding representations makes it apparent that Siamese Networks have indeed incorporated visual semantic information, needed to perform chromosome classification.

5. Conclusions and Future Work

We started by motivating the task of automatic chromosome classification, a necessary and critical step for karyotyping. Further, we have collected a dataset of real world healthy patient chromosomes to carry our experimentation. Next, considering the practical settings where data is limited, we have proposed Siamese Networks for chromosome classification. We demonstrate that a vanilla Siamese Network surpasses the performance of multiple strong baselines created using Deep CNN. In addition to that, we

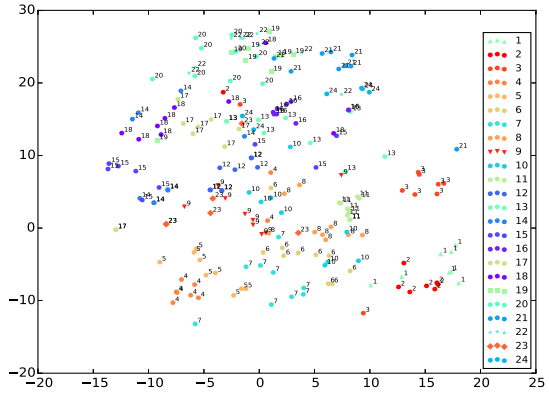


Figure 6. Visualization of embeddings for chromosomes in the test set obtained after reducing their dimensionality to two, using t-SNE [22] method, on the representations learned by the model corresponding to row 15 of Table 1. Figure best viewed magnified.

suggested simple and yet effective methods to augment Siamese Networks thereby expediting their training and prediction steps by *83 and 3 folds* respectively, while maintaining the superior performance accomplished by vanilla Siamese Network.

Looking ahead, we believe that it will be interesting to train an ensemble of Siamese Networks, such that each Siamese Network learns a behaviour that is complementary to the remaining networks of the ensemble. The results obtained by concatenating embeddings from multiple Siamese Networks are not very encouraging, as we have not enforced these networks to be complementary, through any of the available training methods. We would like to emphasize that such an ensemble will not be hampered by training duration as base networks can be parallelized trivially.

References

[1] F. Abid and L. Hamami. A survey of neural network based automated systems for human chromosome classification. *Artificial Intelligence Review*, pages 1–16, 2016. 2

[2] G. Agam and I. Dinstein. Geometric separation of partially overlapping nonrigid objects applied to automatic chromosome classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:1212–1222, 1997. 2

[3] S. Albarqouni, C. Baur, F. Achilles, V. Belagiannis, S. Demirci, and N. Navab. Aggnet: Deep learning from crowds for mitosis detection in breast cancer histology images. *IEEE Transactions on Medical Imaging*, 35(5):1313–1321, 2016. 2

[4] V. S. Balaji and S. Vidhya. Separation of touching and overlapped human chromosome images. *Advancements of Medical Electronics*, pages 59–65, 2015. 2

[5] A. P. Britto and G. Ravindran. A review of cytogenetics and its automation. *Journal of Medical Science*, 7:1–18, 2007. 1,

2

[6] A. Carothers and J. Piper. Computer-aided classification of human chromosomes: a review. *Statistics and Computing*, 4(3):161–171, Sep 1994. 2

[7] F. Chollet. keras. https://github.com/fchollet/keras, 2015. 7

[8] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 539–546, 2005. 6

[9] P. A. Errington and J. Graham. Application of artificial neural networks to chromosome classification. *Cytometry Part A*, 14(6):627–639, 1993. 3

[10] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006. 2, 3

[11] J. Gao, B. Ding, W. Fan, J. Han, and S. Y. Philip. Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing*, 12(6), 2008. 6

[12] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993. 4

[13] H. Irshad, L. Montaser-Kouhsari, G. Waltz, O. Bucar, J. Nowak, F. Dong, N. Knoblauch, and A. Beck. Crowdsourcing image annotation for nucleus detection and segmentation in computational pathology: Evaluating experts, automated methods, and the crowd. *Pacific Symposium on Biocomputing*, pages 294–305, 2014. 2

[14] S. Jahani and S. K. Setarehdan. A novel method for centromere and length detection in microscopic images of human chromosomes. *18th Iranian Conference of Biomedical Engineering*, pages 274–277, 2011. 2, 4

[15] S. Jahani and S. K. Setarehdan. An automatic algorithm for identification and straightening images of curved human chromosomes. *Biomedical Engineering: Applications, Basis and Communications*, 24(06):503–511, 2012. 2, 3, 4

[16] M. Javan-Roshtkhari and S. K. Setarehdan. A new approach to automatic classification of the curved chromosomes. In *5th International Symposium on Image and Signal Processing and Analysis*, pages 19–24. IEEE, 2007. 2

[17] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015. 2, 3

[18] A. Kovashka, O. Russakovsky, L. Fei-Fei, and K. Grauman. Crowdsourcing in computer vision. *arXiv preprint arXiv:1611.02145*, 2016. 2

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 1, 3

[20] B. Lerner, M. Levinstein, B. Rosenberg, H. Guterman, L. Dinstein, and Y. Romem. Feature selection and chromosome classification using a multilayer perceptron neural network. In *IEEE International Conference on Neural Networks*, volume 6, pages 3540–3545, 1994. 3

[21] D. G. Lowe. Object recognition from local scale-invariant features. In *The proceedings of the seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157. Ieee, 1999. 3

[22] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 8, 9

[23] N. Madian and K. Jayanthi. Overlapped chromosome segmentation and separation of touching chromosome for automated chromosome classification. In *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5392–5395, 2012. 2

[24] J. Piper. Automated cytogenetics in the study of mutagenesis and cancer. *Advances in Mutagenesis Research*, pages 127–153, 1990. 1

[25] J. Piper, E. Granum, D. Rutovitz, and H. Ruttledge. Automation of chromosome analysis. *Signal Processing*, 2(3):203–221, 1980. 2

[26] S. Prakash and N. K. Chaudhury. Dicentric chromosome image classification using fourier domain based shape descriptors and support vector machine. In *Proceedings of International Conference on Computer Vision and Image Processing*, pages 221–227. Springer, 2017. 2

[27] M. J. Roshtkhari and S. K. Setarehdan. A novel algorithm for straightening highly curved images of human chromosome. *Pattern Recognition Letters*, 29(9):1208 – 1217, 2008. 2, 3, 4

[28] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016. 3

[29] D. Schlesinger, F. Jug, G. Myers, C. Rother, and D. Kainmüller. Crowd sourcing image segmentation with iastaple. *arXiv preprint arXiv:1702.06461*, 2017. 2

[30] M. Sharma, O. Saha, A. Sriraman, L. Vig, R. Hebbalaguppe, and S. Karande. Crowdsourcing for chromosome segmentation and deep classification. In *CVPR 2017*. IEEE CVPR 2017, 2017. 2, 3, 4

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 3

[32] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017. 2

[33] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008. 2, 4

[34] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 7

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 1, 3

[36] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016. 7

[37] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016. 1, 3

[38] X. Wang, B. Zheng, S. Li, J. J. Mulvihill, M. C. Wood, and H. Liu. Automated classification of metaphase chromosomes: Optimization of an adaptive computerized scheme. *Journal of Biomedical Informatics*, 42(1):22–31, 2009. 1