

UNIVERSITY OF PADOVA

DEPARTMENT OF INFORMATION ENGINEERING

BSc IN COMPUTER ENGINEERING

Deep Learning-Based Human Chromosome Classification: Data Augmentation and Ensemble

Supervisor:

PROF. LORIS NANNI

Author:

MATTIA D'ANGELO

July 19, 2023

Academic Year 2022/2023

Abstract

Object classification is a crucial task in deep learning, which involves the identification and categorization of objects in images or videos. Although humans can easily recognize common objects, such as cars, animals, or plants, performing this task on a large scale can be time-consuming and error-prone. Therefore, automating this process using neural networks can save time and effort while achieving higher accuracy. Our study focuses on the classification step of human chromosome karyotyping, an important medical procedure that helps diagnose genetic disorders. Traditionally, this task is performed manually by expert cytologists, which is a time-consuming process that requires specialized medical skills. Therefore, automating it through deep learning can be immensely useful. To accomplish this, we implemented and adapted existing preprocessing and data augmentation techniques to prepare the chromosome images for classification. We used ResNet-50 convolutional neural network, and Swin Transformer, coupled with an ensemble approach to classify the chromosomes, obtaining state-of-the-art performance in the tested dataset.

Abstract (italiano)

La classificazione di oggetti è un compito cruciale nel campo del deep learning, che ne comporta l'identificazione e la categorizzazione in immagini o video. Sebbene gli esseri umani possano facilmente riconoscere oggetti comuni, come automobili, animali o piante, svolgere questo compito su larga scala può essere dispendioso in termini di tempo e soggetto ad errori. Pertanto, automatizzare questo processo utilizzando le reti neurali può far risparmiare tempo e fatica, garantendo al contempo una maggiore precisione. Il nostro studio si concentra sulla fase di classificazione del cariotipo umano, un'importante procedura medica che aiuta a diagnosticare disturbi genetici. Tradizionalmente, questo compito viene svolto manualmente da citologi esperti, un processo che richiede molto tempo e competenze mediche specifiche. Pertanto, automatizzarlo attraverso il deep learning può essere estremamente utile. Per raggiungere questo obiettivo, abbiamo implementato e adattato tecniche esistenti di pre-elaborazione e di aumento dei dati in modo da preparare le immagini dei cromosomi per la classificazione. Quindi, abbiamo utilizzato reti neurali convoluzionali di tipo ResNet-50 e lo Swin Transformer, assieme a tecniche di ensemble per classificare i cromosomi, ottenendo risultati all'avanguardia sul dataset utilizzato.

Contents

1	Introduction	1
2	Materials and Methods	5
2.1	A brief introduction to CNNs	5
2.1.1	ResNet-50	6
2.2	A brief introduction to Transformers	6
2.2.1	Swin	7
2.3	The dataset	7
2.4	Data Augmentation	8
2.5	Straightening	10
2.5.1	Chromosome image binarization and bending centre locating	11
2.5.2	Chromosome arms straightening and final image creation	13
2.5.3	Algorithm improvements	15
2.5.4	Additional tests	16
2.6	Feature Transform-based techniques	17
3	Results	21
3.1	Metrics	21
3.2	Experiment settings	22
3.3	Experimental results	22
4	Conclusions	25
	References	27

Chapter 1

Introduction

Karyotyping is the laboratory medical procedure that allows to individuate the karyotype, which is an organism's complete set of chromosomes. Studying the size, number, and shape of chromosomes is important to diagnose cancers and genetic disorders, such as monosomy, trisomy or deletions at an early stage [1].

Chromosomes consist of DNA molecules containing genetic information. In a healthy human cell there are 46 chromosomes [2], organized into 23 pairs. They include 22 pairs of autosomes and a pair of sex chromosomes (X and Y in male cells and double X in female cells). In order to study them, cytologists perform karyotyping by first taking cells from the patient (specifically from peripheral blood, amniotic liquid or bone marrow) and culturing them in vitro. During the metaphase stage of mitosis, chromosomes become distinguishable and hence the cells are arrested. Subsequently, a staining technique is employed to highlight the morphological features of the chromosomes, such as bands, by applying a specific dye. One commonly used staining procedure is G banding, which utilizes Giemsa staining to color the chromosomes, enhancing dark and light bands, length and centromere position.

After the staining, a chromosome picture is taken through a microscope, producing a micrograph. Cytologists visually inspect this image, examining chromosome features such as size, shape, banding pattern, centromere position. Referring to the human idiogram published by the International System for Human Cytogenomic Nomenclature (ISCN), the specialists can organize and pair chromosomes according to these characteristics; also, classes are assigned to each pair: there are 24 classes for males and 23 for females (because there is no Y chromosome) [3]. The result of the karyotyping is a graphical representation called a karyogram, illustrated in Figure 1.

While karyotyping is an essential procedure, it is also time-consuming and requires skilled cytologists to work manually on cells. As a result, researchers have developed computational techniques [4, 5, 6, 7] to automate it, often adopting artificial neural networks to save time and effort.

The two main challenges of automated karyotyping are chromosome segmentation and chromosome classification. The first replaces the manual metaphase detection, a

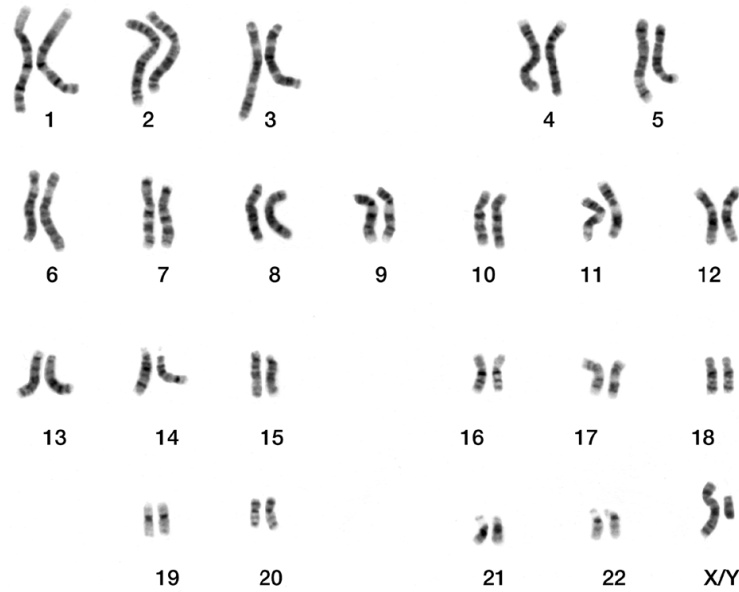


Figure 1: Micrographic karyogram of human male using Giemsa staining. The numbers indicate the classes of chromosomes.

procedure which requires cytologists to examine hundreds to thousands of metaphase images, most of which contains debris and noises. The images are classified into analysable and unanalysable according to the chromosome features, and only analysable ones are used for karyotyping [3]. Artificial neural networks can recognize the analysable images and then extract from them, through segmentation techniques, the chromosome instances. These are randomly located within the image and often overlap and adhere together. By defining chromosome geometric features such as area, bounding box, axis length and using specific algorithms, including those based on machine learning, it is possible to extract the chromosomes from the images and subsequently classify them [8].

The second important stage in the karyotyping procedure is the classification of the chromosome instances into 24 different types or classes, i.e., they are labeled from 1 to 22 and X, Y. Similar to segmentation, this process can be automated through convolutional neural networks (CNNs) trained on large datasets of labeled chromosome images. However, collecting such datasets from medical institutions can be challenging due to privacy-related constraints. Additionally, due to their non-rigid body, chromosomes can be bent or oriented in different ways, which makes their classification more difficult [9]. Differences in cell cultivation techniques and micrograph productions increase the problem [10]. To address these issues, data augmentation and straightening techniques can be employed. The first one artificially produces more data samples starting from a base dataset (images are modified by applying scaling, rotation, noise addition, blurring, or other transformations), while the second can straighten bent chromosomes. These procedures can achieve higher classification accuracy and are applied to the original chromosome images before training the neural network for classification.

Our work focused on chromosome classification. We implemented a data augmenta-

tion technique proposed by Lin et al. [11], as well as a common straightening procedure [12] improved by Sharma et al. [13]. We introduced some modifications and improvements to the baseline methods. Finally, we applied three different image processing techniques based on the Features Transform (Fourier and Discrete cosine). The combination of them, and the use of a ResNet-50/SWIN ensemble for the classification, show better results than previous works.

The paper is organized as follows: in Section 2 we present the dataset used in this work, as well as the methods used for data augmentation and classification. In Section 3, we describe the performed experiments; moreover, we discuss the reported performance. In Section 4 the conclusion and some proposals for future research are given.

The MATLAB source code of the methods here reported is freely available at:
<https://github.com/MattiaDAngelo>.

Chapter 2

Materials and Methods

This section provides a complete description of the dataset used and the image processing techniques mentioned above. In particular, Section 2.1 introduces convolutional neural networks to better understand the context of this work, Section 2.2 introduces SWIN transformer, Section 2.3 describes the dataset, Section 2.4 describes the data augmentation technique, Section 2.5 describes the straightening procedure and Section 2.6 describes the feature transform-based techniques.

2.1 A brief introduction to CNNs

Convolutional Neural Networks (CNNs) are powerful deep learning models commonly used for image detection and classification tasks. They are designed to replicate how our brain recognizes images, by extracting relevant features from them.

One of the key advantages of CNNs is their ability to be invariant to certain changes in the image. For example, a CNN can still recognize a cat even if the image is resized, rotated, or exposed to different lighting conditions if it has learned what features are important for identifying a cat. To achieve this level of performance, CNNs are trained on large datasets of images, with multiple samples of the same object to ensure that the network learns the relevant features and can generalize to new, unseen, images.

A CNN is typically composed of some convolutional blocks used for feature extraction, followed by fully connected layers used for classification. The network input is either a grayscale image or an RGB image and the output is the class prediction.

A convolutional block is composed of one or more convolutional layers and a pooling layer. Convolutional layers apply one or more filters to their input, performing a convolution operation. This allows the network to learn relevant features from the image, producing a feature map. After one or more convolutional layers, it is common to include a pooling layer: its purpose is to reduce the spatial size of the feature maps, which also reduces the number of parameters and computation required in the network.

Finally, fully connected layers classify the image by mapping the learned features to the output classes, i.e., they produce an output vector representing the predicted class

probabilities for the input image.

2.1.1 ResNet-50

The ResNet-50 is a type of CNN architecture with 50 layers, introduced in 2015 by a team of researchers at Microsoft Research [14].

In general, residual neural networks (ResNet) were introduced to solve the “vanishing gradient problem”, a phenomenon that can occur during the training of deep neural networks with many layers and makes it difficult to obtain accurate predictions.

ResNet-50 has been pretrained on the large ImageNet dataset and therefore can classify objects up to 1000 different categories [15].

2.2 A brief introduction to Transformers

Transformers are deep learning models used in the fields of natural language processing (NLP), computer vision, and audio processing. First introduced in 2017 by Vaswani et al. [16] they are now the most commonly used models for NLP problems [17] and can also be applied to image classification tasks [18].

Transformers basically consist of a stack of encoders and a stack of decoders: the encoding component takes a sentence to translate as input, tokenizes it, and applies attention. The decoding component takes the result and outputs the sentence in a different language.

The encoders consist of two sub-layers: self-attention and a feed-forward neural network. Self-attention is key because it allows the transformer to “understand” the structure of the sentence and how the words are related to each other. The first step is embedding the words of the sentence, i.e., turning them into vectors. The embeddings are packed into a matrix and multiplied by some weight matrices to produce three more matrices called query, key, and value. Then, it is possible to apply Equation (1) representing a normalized matrix as the output of the self-attention layer. This expresses the “similarity” of the words to each other. The result is moved to the feed-forward neural network, allowing the model to capture more complex relationships between the words.

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{D_k}}\right)V \quad (1)$$

In Equation (1), $Q \in R^{N \times D_k}$ is the query matrix, $K \in R^{M \times D_k}$ is the key matrix, $V \in R^{M \times D_v}$ is the value matrix, D_k is the dimension of the keys, N is the length of the queries and M is the length of the keys.

The decoders have both the self-attention layer and the feed-forward network, along with an attention layer in the middle. The decoders produce new tokens based on the encoder’s output until reaching an end-of-sentence token. The output is normalized, producing probabilities associated with each word, and words with the highest probabilities

are chosen, representing the translation of the corresponding input words.

Transformers have also been used for image classification. The ViT model [18] was the first transformer to achieve results equivalent to or better than CNNs in image classification.

2.2.1 Swin

Swin [19] is another transformer model that has reached state-of-the-art performance in vision tasks. It was inspired by the ViT model, introducing hierarchical feature maps and shifted windows to improve performance and execution time. The SWIN pre-trained model from the Timm library (<https://timm.fast.ai/>) is used in this work, with a patch size of 4. To mitigate overfitting, we implemented a validation split, reserving 1/4 of the training set for validation purposes. Additionally, we resized the image dimensions to 224×224 to match the required input size.

During the training process, we employed the AdamW optimizer with cosine annealing. To maintain network stability, we utilized a low learning rate of 10^{-4} and applied a weight decay of 0.03. Our chosen batch size was 32, and for the loss function, we employed the standard cross-entropy.

In order to enhance convergence, we monitored the minimum validation F1 score over consecutive epochs. If no decrease was observed after five epochs, we further reduced the learning rate to 10^{-5} and subsequently to 10^{-6} . This adjustment aimed to improve model performance. Ultimately, we saved the best-performing model, the one with the highest F1 score during validation throughout the training process.

2.3 The dataset

The dataset used in this study was obtained from Lin et al.'s GitHub repository (<https://github.com/CloudDataLab/CIR-Net>) and it contains 2986 G-band chromosome images. They represent 65 different karyotypes, of which 32 are male and 33 are female; in particular, the dataset consists of 130 chromosomes per label from 1 to 22, except for labels 8, 12, 16, and 19, which are associated with only 129 chromosomes. Sex chromosomes are composed of 98 type X chromosomes labeled to 23, and 32 type Y chromosomes labeled to 24. The images are grayscale with 8-bit color depth, which means pixels have an intensity range from 0 (black) to 255 (white). The size is 224x224 pixels, with the background being black and the chromosome composed of shades of gray. Figure 2 (a) illustrates an image of a type 8 chromosome from the dataset.

We also looked for other datasets that might be useful for testing purposes. For convenience, we report here the papers and links to each of the datasets found: BioImLab dataset for segmentation [20] and BioImLab dataset for classification [21]; Passau Chromosome Image Data (Pki-3) [22]; Guangdong Women and Children Hospital dataset [23]. The first dataset contains 162 PAL-resolution quinacrine (Q)-banded prometaphase

images and 117 manual karyograms; the second dataset consists of 5.474 Q-band single chromosome images; *Pki-3* contains tiff-images of 612 human metaphase cells and the associated karyograms using Giemsa staining; the *Guangdong Women and Children Hospital dataset* contains tiff-images of 528 chromosome instances and 105 chromosome clusters using Giemsa staining.

2.4 Data Augmentation

Data augmentation refers to a set of techniques used to manipulate and transform existing data in order to artificially increase the number of samples in a dataset. The primary goal of data augmentation is to improve the performance of deep learning models and prevent overfitting. This procedure is commonly applied to image datasets, where it involves using various methods such as geometric transformations, color space augmentations, and random erasing [24].

The dataset described in Section 2.3 is relatively small and will bring to a not accurate classification. In their work, Lin et al. [11] proposed a data augmentation algorithm called *CDA* (Chromosome Data Augmentation), which generates additional data samples from the original dataset: the algorithm introduces random variations in the position and orientation of chromosomes within the images.

The key step of the *CDA* algorithm is to apply affine transformations to the original chromosome images, which are grayscale. In computing, a grayscale image is represented as a matrix of pixels. Its size is expressed as height and width, which are the number of rows and columns of pixels, respectively. Affine transformations are commonly used on such images to scale, rotate, or reflect them.

In the *CDA* algorithm, the images are rotated by a specific angle θ and then translated by a randomly determined offset. Equation (2) formalizes the process:

$$x' = A(\theta)x + b \quad (2)$$

$$A(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (3)$$

$$b = \begin{bmatrix} r_{offset} \\ c_{offset} \end{bmatrix} \quad (4)$$

In Equation (2), x denotes the original image; $A(\theta)$ is a rotation matrix formalized in Equation (3); b is a translation vector formalized in Equation (4), where r_{offset} and c_{offset} denote the pixel offset in rows and columns. x' denotes the augmented image.

As said before, the translation offsets are randomly determined. It's important to choose them so the chromosomes are always within the boundaries of the images, or there would be a loss of pixels. Therefore, we used the built-in MATLAB function *regionprops*, which can return the position and the size of the smallest box containing the region, i.e.,

the chromosome. Using this information, we can constrain the offsets to a specific range so that shifting the chromosome horizontally or vertically ensures that it remains within the image boundaries.

In some cases, multiple bounding boxes are returned for certain images in the dataset due to the presence of artifacts such as non-black pixels outside the chromosome region. To address this, we select only the largest bounding box from the set of bounding boxes, as it corresponds to the rectangle that encloses the chromosome region.

The *regionprops* function requires a binary image that consists of pixels either 0 (black) or 1 (white). In the dataset used, common binarization techniques like Otsu's method did not yield satisfactory results. This is because some parts of the chromosome, such as the bands, have similar shades of color to the black background, leading to loss of chromosome borders as they are mistaken for the background. Therefore, we opted for a simple binarization method where every pixel with a value greater than 0 is set to 1. This approach separates the chromosome well from the background, coloring it white and making it possible to apply the *regionprops* function (Figure 2 (b)). Note that if background pixel values are modified by other image processing techniques, this binarization method may not be optimal. In such cases, alternative segmentation approaches may be necessary.



Figure 2: (a) Grayscale image of a type 8 chromosome and (b) its binary version, including the chromosome bounding box.

Algorithm 1 describes the data augmentation operation.

Algorithm 1 Chromosome Data Augmentation

Input: A chromosome image x

The chromosome label y corresponding to x

Output: The set of augmented images

```

1:  $augmented\_set \leftarrow \{\}$ 
2: for  $\theta \leftarrow 15$  to 345 step 15 do
3:    $x_\theta \leftarrow A(\theta)x$ 
4:    $x_{bw} \leftarrow \text{binarize } x_\theta$ 
5:    $bounding\_box \leftarrow \text{find largest chromosome bounding box using } x_{bw}$ 
6:    $r_{offset} \leftarrow \text{calculate random vertical offset according to } bounding\_box$ 
7:    $c_{offset} \leftarrow \text{calculate random horizontal offset according to } bounding\_box$ 
8:    $b \leftarrow \begin{bmatrix} r_{offset} \\ c_{offset} \end{bmatrix}$ 
9:    $x' \leftarrow x_\theta + b$ 
10:   $augmented\_set \leftarrow augmented\_set \cup \{(x', y)\}$ 
11: end for
12: return  $augmented\_set$ 

```

We apply Algorithm 1 for each image in the dataset and we generate a set composed of original and augmented images.

As specified by Lin et al., we rotate each chromosome by 15 to 345 degrees in steps of 15 degrees. Some examples are shown in Figure 3. As a result, each image generates a total of 23 augmented images. The data augmentation is applied only to the training images.

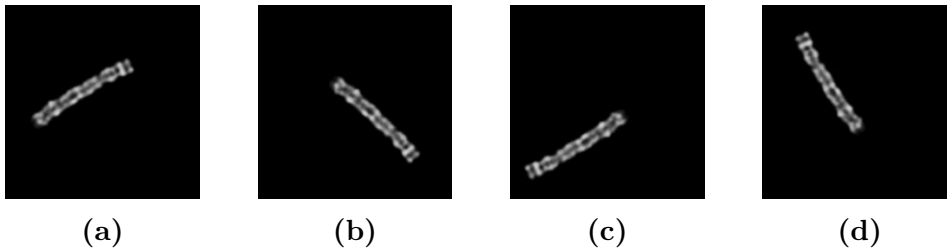


Figure 3: Chromosome images rotated clockwise by (a) 60°, (b) 135°, (c) 240°, (d) 330° and horizontally and vertically randomly shifted.

2.5 Straightening

The straightening procedure is a preprocessing step (i.e., something done before training the neural network) aimed at removing the random curves and orientation of a chromosome by making it straight and vertical within the image. This allows the neural network classifier to focus on other important features and refrain from learning to discriminate based on curves and orientation, as seen in the *CDA* algorithm.

We implemented in MATLAB a known straightening algorithm proposed in 2008 [12], with improvements from a 2017 work [13], and some personal adaptations for improving it.

The algorithm finds the bending centre of a curved chromosome, i.e., the point where the latter is bent, and uses this information to straighten the chromosome. Here we provide the description of the procedure.

2.5.1 Chromosome image binarization and bending centre locating

First, the chromosome grayscale image needs to be binarized. For the dataset in use, we adopted the same binarization technique described in the data augmentation section, so all the pixels greater than 0 are set to 1. The resulting binary image has a black background and a white chromosome, as shown in Figure 4. Then, it is cropped to retain only the chromosome.



Figure 4: (a) Grayscale image of a type 5 chromosome and (b) its binary version.

Next, the algorithm calculates the horizontal projection vector of the binary image by summing up the pixel values on each row. The horizontal projection contains the morphological information of the chromosome. Analyzing its extrema, it is possible to find the bending centre as explained below (to minimize the effects of small deflections on the location of the extrema, we applied a *Savitzky-Golay* filter using the built-in MATLAB function *sgolayfilt*, which smooths out the horizontal projection).

For a straight chromosome, the global minimum point in the horizontal projection vector corresponds to the centromere, the thinner part of a chromosome [25]. This point often corresponds to the bending centre of a curved chromosome. However, since the horizontal projection vector is strongly dependent on the position of the chromosome and its degree of bending, locating just the global minimum point would not be effective. To overcome this, the algorithm rotates the binary image from 0° to 180° by steps of 10° and analyzes the corresponding projection vectors. It is found that the actual bending centre corresponds to the global minimum point between two locally global maxima with similar amplitudes. Since there is a projection vector for each rotated image, the correct bending centre is selected in the image that presents the minimum *rotation score* S , defined in Equation (5):

$$S = w_1 \times R_1 + w_2 \times R_2 \quad (5)$$

$$R_1 = \frac{|P_1 - P_2|}{P_1 + P_2} \quad (6)$$

$$R_2 = \frac{P_3}{P_1 + P_2} \quad (7)$$

In Equations (6) and (7), P_1 and P_2 represent the values of the two locally global maxima, while P_3 represents the value of the global minimum located between them. R_1 describes the relation between the two maxima: if they have similar amplitudes, then R_1 will be smaller. R_2 describes the relative amplitude of the global minimum between the two maxima. In Equation (5), w_1 and w_2 are the tuning parameters whose selection influences the location of the bending centre. The location of the global minimum point P_3 in the horizontal projection vector identifies the bending axis of the chromosome, i.e., a horizontal line which divides the chromosome into two arms; the outermost point of intersection between the chromosome and the bending axis is the actual bending centre of the chromosome. The opposite point, i.e., the last chromosome pixel along the same bending axis with respect to the bending centre, we called *unjoined point* and will be used in a further process. An idea of the process explained above is presented in Algorithm 2.

Algorithm 2 Rotation Score Calculation

Input: A chromosome grayscale image x and its binary version x_{bw}

The tuning parameters w_1, w_2

Output: The rotated grayscale image x'

The rotated binary image x'_{bw}

The *index* of the global minimum in the horizontal projection vector

```

1:  $S \leftarrow Inf$ 
2:  $index \leftarrow 0$ 
3:  $\theta_S \leftarrow 0$ 
4: for  $\theta \leftarrow 0$  to 180 step 10 do
5:    $x'_{bw} \leftarrow A(\theta)x_{bw}$ 
6:   horizontal_projection  $\leftarrow$  calculate the horizontal projection vector of  $x'_{bw}$ 
7:   horizontal_projection_smoothed  $\leftarrow$  apply a Savitzky-Golay filter to
   horizontal_projection
8:    $P_3 \leftarrow$  calculate the global minimum
9:    $P_1, P_2 \leftarrow$  calculate the two global maxima around  $P_3$ 
10:   $S_\theta \leftarrow w_1 \times R_1 + w_2 \times R_2$ 
11:  if  $S_\theta < S$  then
12:     $S \leftarrow S_\theta$ 
13:     $index \leftarrow$  index of  $P_3$  inside horizontal_projection_smoothed
14:     $\theta_S \leftarrow \theta$ 
15:  end if
16: end for
17:  $x' \leftarrow A(\theta_S)x$ 
18:  $x'_{bw} \leftarrow A(\theta_S)x_{bw}$ 
19: return  $x', x'_{bw}, index$ 

```

Figure 5 illustrates the smoothed horizontal projection, the corresponding bending axis and the bending centre. The chromosome shown has been rotated counterclockwise by 20° , which corresponds to the lowest *rotation score* S in this case.

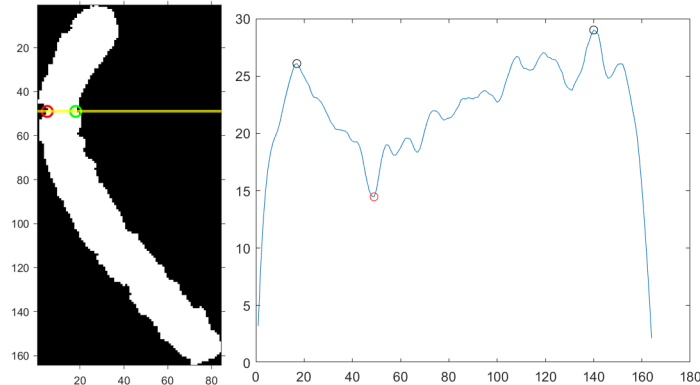


Figure 5: From the left: type 5 chromosome image rotated counterclockwise by 20° (yellow line: bending axis; red circle: bending centre; green circle: unjoined point), the corresponding smoothed horizontal projection (black circles: maxima points; red circle: minimum point identifying the bending axis).

A point on the tuning parameters w_1 , w_2 described in Equation 5: these weights affect the performance of the straightening procedure; they can be chosen by visually analyzing the bending axis in the chromosome images. We found that setting w_1 to 0.67 and w_2 to 0.33 produced good results.

2.5.2 Chromosome arms straightening and final image creation

The chromosome image is now divided along the bending axis into two sub-images, each containing an arm of the chromosome (upper and lower). Both sub-images are rotated from -90° to 90° by steps of 10° using a rotation matrix. For each rotation, the vertical projection vector is calculated by summing up the pixel values on each column. The vertical projection vector with minimum width is associated with the arm in a vertical position within the image (see Algorithm 3). The coordinates of the bending centre and the unjoined point in both sub-images are recalculated in the vertical image after the rotation.

As can be seen in Algorithm 3, the operation is applied to both the binary image and the grayscale image. The two sub-images are then cropped to retain only the chromosome arms; this will be useful in the alignment procedure. Figure 6 shows the result.

Finally, the two arms must be aligned and connected. Thinking of the image pixels to be in a Cartesian plane, with the origin in the upper left corner and the horizontal axis toward the right, we compute the difference between the x-coordinate of the bending centre in the upper and lower arm, resulting in a horizontal shift called *shift x*. Then, we need to move to the right the image with the leftmost bending centre: to avoid losing arm pixels, the selected image's right border is padded with $|\text{shift } x|$ black pixel columns and it is moved by $|\text{shift } x|$. This ensures that the two arms are perfectly aligned along

Algorithm 3 Chromosome Arm Straightening

Input: A chromosome arm grayscale image x and its binary version x_{bw}

Output: The grayscale straightened arm x'

The binary straightened arm x'_{bw}

```

1:  $width \leftarrow Inf$ 
2:  $\alpha \leftarrow 0$ 
3: for  $\theta \leftarrow -90$  to  $90$  step  $10$  do
4:    $x'_{bw} \leftarrow A(\theta)x_{bw}$ 
5:    $vertical\_projection \leftarrow$  calculate the vertical projection vector of  $x'_{bw}$ 
6:    $width_\theta \leftarrow$  calculate the number of non-zero elements in  $vertical\_projection$ 
7:   if  $width_\theta \leq width$  then
8:      $width \leftarrow width_\theta$ 
9:      $\alpha \leftarrow \theta$ 
10:  end if
11: end for
12:  $x' \leftarrow A(\alpha)x$ 
13:  $x'_{bw} \leftarrow A(\alpha)x_{bw}$ 
14: return  $x', x'_{bw}$ 

```

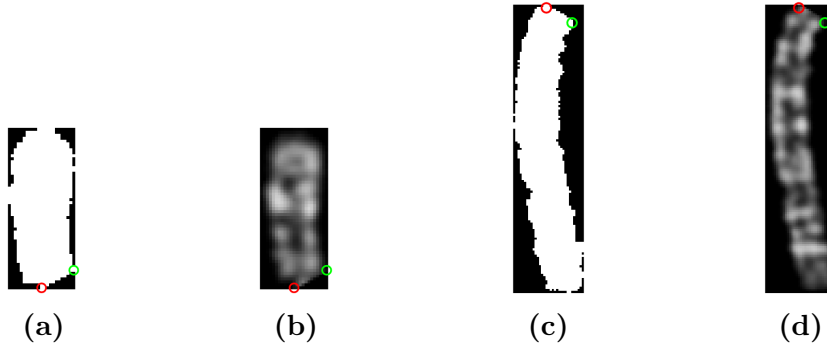


Figure 6: (a), (b) Straightened upper arm and (c), (d) lower arm. Red circle: bending centre; green circle: unjoined point.

the x-coordinate of the bending centre. The coordinates of the points are then updated to always refer to the same pixels.

Now, we can just copy all the pixels of the two sub-images in a new image that corresponds to the straightened version of the original chromosome. Obviously, in the upper part of the image there will be the upper arm, and in the lower part of the image there will be the lower arm (Figure 7).

The last step of the algorithm, proposed by Sharma et al. [13], involves reconstructing the area of the chromosome lost during the straightening. This is performed on the grayscale image by connecting the unjoined points through a straight line and filling the enclosed area with the mean value of the non-black pixels at the same horizontal level. This reconstructs the missing part and preserves the horizontal bands of the chromosome. However, this procedure doesn't give optimal results with the binarization technique we used: chromosome border pixels are dark but not completely black and therefore are included in the binarization; when reconstructing the missing area, they lower the mean resulting in an overall color that is darker than it should be; moreover, they result in a



Figure 7: (a) Binary and (b) grayscale image of the straightened chromosome.

non-homogeneous shade of color, as shown in Figure 8 (c). To produce a more uniform color, each pixel value in the area enclosed by the bending centre points and the unjoined points (upper and lower) is replaced with the mean of the surrounding non-black pixels (using a radius of 1), resulting in a smoother color that still preserves the shade of the chromosome bands (Figure 8 (b)).

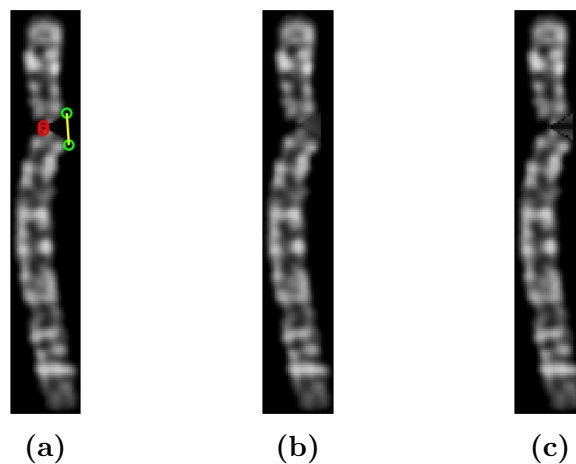


Figure 8: (a) The straightened chromosome image with reference points (red and green circles). (b) The corresponding filled version. (c) The same chromosome but using the mean value of the non-black pixels to reconstruct the missing area.

Finally, the algorithm pads the borders of the straightened image with black pixels to resize it to the same size as the original image, before returning it as the final output, illustrated in Figure 9.

2.5.3 Algorithm improvements

The effectiveness of the straightening procedure can be enhanced by applying it only to the bent chromosomes. Since the upright tightest fitting rectangle for a straight chromosome contains fewer black pixels compared to a bent one, it is possible to automatically determine whether a chromosome is bent or straight. We applied this idea by manually



Figure 9: The final straightened chromosome image.

selecting 1014 visibly straightest chromosomes from the original dataset of 2986 images, i.e., those chromosomes with arms forming an angle of approximately 180° . Then, we binarized each image and calculated the *whiteness value* W [13], defined as the ratio of the sum of white pixels and the area of the tightest fitting rectangle of the chromosome. The closer W is to 1, the straighter the chromosome is. We used the MATLAB function *regionprops* to find the rectangle, i.e., the bounding box. We selected a *whiteness threshold* W_T of 0.667, meaning that only chromosomes with $W < W_T$ are considered bent and therefore are sent for further straightening. Using this method, we identified 1472 bent chromosomes from the original dataset.

To improve classification accuracy, we added some check conditions to the straightening algorithm: not all bent chromosomes can be effectively straightened due to their particular shapes and the effects of the weights w_1 and w_2 in Equation 5. In general, we consider an effective straightening to be one in which the chromosome arms are vertical and oriented correctly (i.e., not upside down). Thus, we check the relative location of the bending center and the unjoined point in both the upper and lower arm images: if these conditions are not met, the algorithm stops and returns the original chromosome image. This approach avoids incorrectly straightening 447 chromosomes, resulting in a more robust dataset composed of original and well-straightened images.

2.5.4 Additional tests

We evaluated the effectiveness of the proposed straightening technique by applying it to some chromosome images from two different datasets [20, 22]. The first dataset contains Q-band chromosome images with a black background, while the second contains G-band chromosome images with a white background. As the binarization technique described in the previous sections works better with black backgrounds, we modified it to handle white backgrounds as well. We achieved this by computing the complement of the white background image, which turns the background black. We apply the straightening procedure to this modified image and finally, we restore its original color by computing the complement again. The straightening procedure worked well on the chromosome images from both datasets, as shown in Figures 10 and 11.

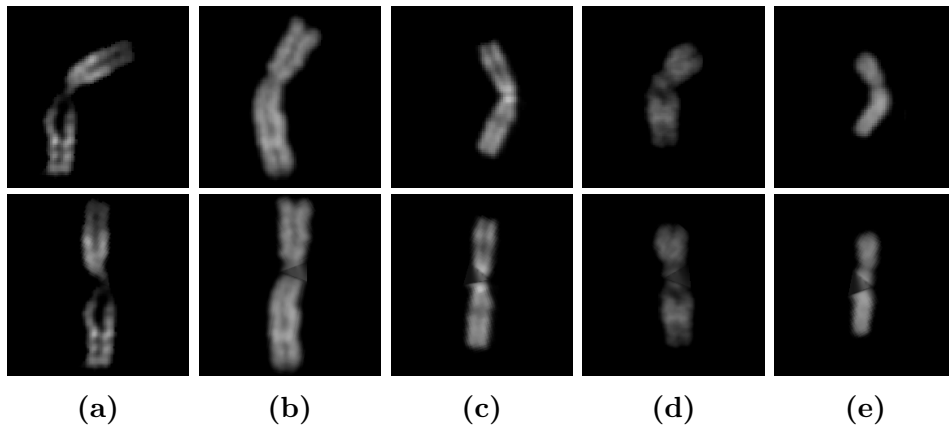


Figure 10: Type (a) 1, (b) 2, (c) 3, (d) 11, (e) 23 Q-band chromosome images and their straightened version.

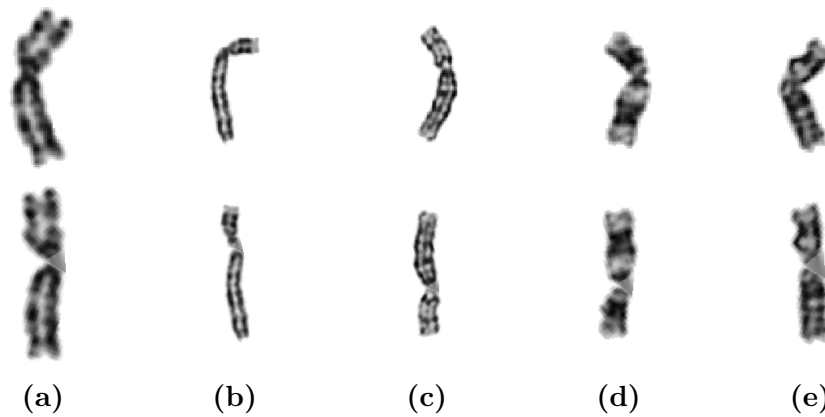


Figure 11: Type (a) 3, (b) 4, (c) 6, (d) 11, (e) 23 G-band chromosome images and their straightened version.

2.6 Feature Transform-based techniques

To enhance the contrast, blur, and add noise to the chromosome images, three image processing methods are utilized. These involve the use of a *2-D Fast Fourier Transform* (FFT) for the first two techniques and a *2-D Discrete Cosine Transform* (DCT) for the third one. For each image, three new images are created.

These techniques are applied to grayscale images. If necessary, input RGB images are converted to grayscale and restored to RGB after processing. The methods are described in the following steps:

I. First technique

1. Apply FFT to the grayscale chromosome image.
2. Shift the zero frequency component to the center of the frequency-domain.
3. Create a mask made of 1s with the same dimensions as the transformed and frequency-shifted image; this will preserve only selected frequencies in step 7.
4. Create two 2-D grids to represent the x and y coordinates.

5. Use the grids to compute the euclidean distance and get a colored distance matrix R (see Figure 12).
6. Select a radius (threshold) $r = 2$ and set to zero the values of the mask inside this radius using the values of distance matrix R as coordinates.
7. Apply the mask to the transformed image.
8. Apply the inverse Fourier Transform (iFFT) to get the blurred grayscale image, visible in Figure 13 (b).

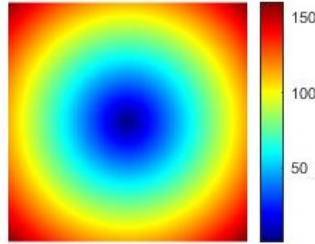


Figure 12: The distance matrix R .

II. Second technique

This technique is similar to the previous one, except that the zero frequency is not shifted:

1. Apply FFT to the grayscale image.
2. Sort and store in an array the elements (IDs) of the transformed image by their intensity values.
3. Select a value $p = 0.5$ that represents the percentage of points in the transformed image that will be set to zero.
4. Randomize the IDs array and set a portion of the elements corresponding to the selected percentage p to zero in the transformed image.
5. Apply the inverse Fourier Transform (iFFT) to recover the filtered grayscale image. The result is shown in Figure 13 (c).

III. Third technique

This method uses the Discrete Cosine Transform (DCT) for image processing instead of the Fourier Transform:

1. Convert the input RGB image to grayscale.
2. Apply the DCT to the grayscale image to obtain its frequency components.
3. Set to zero a low frequency range, which is a 10×10 square in this case.
4. Apply the inverse DCT (iDCT) to obtain the processed grayscale image (Figure 13 (d)).

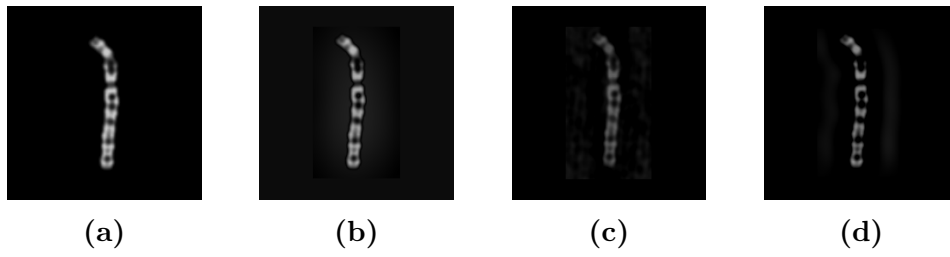


Figure 13: (a) Type 6 chromosome image and the corresponding results after applying (b) first technique, (c) second technique and (d) third technique.

Chapter 3

Results

This section describes the evaluation metrics, the experiment settings and the experimental results on the methods presented.

3.1 Metrics

We evaluated the performance of the ResNet-50 using typical metrics: *precision*, *recall*, *accuracy*, F_1 score (F_1), which can be compared to the ones obtained by other studies. In the context of multi-class classification, *precision* is defined as Equation (8), *recall* as Equation (9), *accuracy* as Equation (10), F_1 as Equation (11).

$$precision = \frac{1}{M} \sum_{i=1}^M \frac{TP_i}{TP_i + FP_i} \quad (8)$$

$$recall = \frac{1}{M} \sum_{i=1}^M \frac{TP_i}{TP_i + FN_i} \quad (9)$$

$$accuracy = \frac{1}{N} \sum_{i=1}^M TP_i \quad (10)$$

$$F_1 = \frac{1}{M} \sum_{i=1}^M \frac{2 \times TP_i}{2 \times TP_i + FN_i + FP_i} \quad (11)$$

In the above equations, TP_i stands for *True Positives*, i.e., chromosomes of type i are correctly classified as type i ; FP_i stands for *False Positives*, i.e., chromosomes of type i are incorrectly classified as type j ; FN_i stands for *False Negatives*, i.e., chromosomes of type i are incorrectly classified as type j ($\forall i \neq j$). M is the number of chromosome types and N is the number of chromosome images in the test set.

3.2 Experiment settings

To classify the chromosomes, we utilized the ResNet-50 pre-trained on ImageNet. In line with transfer learning, we fine-tuned the pre-trained network by replacing its last three layers, responsible for image classification, with a fully connected layer, a softmax layer, and a classification layer adapted to handle the number of chromosome classes. We trained the ResNet-50 network with the following hyperparameters: mini-batch of 30, epochs of 20, and a learning rate of 0.001. The 5-fold cross-validation is used as a test protocol: so we have five different training-test splits: for each split, 80% of the images belong to the training set, 20% to the test set.

For a subset of tests, we also used SWIN; due to computational issues, we used only a subset of the tests reported for ResNet-50.

3.3 Experimental results

This section presents the performance achieved by the ensemble proposed in our study and those from some previous research, all tested on the same dataset with the same testing protocol. The experimental results are presented in Table 3.1.

Table 3.1: Experimental results.

Method	Precision	Recall	Accuracy	F1
<i>Vanilla-CNN</i> [6]	0.8800	0.8600	0.8644	0.8700
<i>SiameseNet</i> [7]	0.8800	0.8700	0.8763	0.8700
<i>CIR-Net</i> [11]	0.9600	0.9600	0.9598	0.9600
ResNet-50				
<i>None(1)</i>	0.8834	0.8767	0.8808	0.8775
<i>None(10)</i>	0.9246	0.9157	0.9216	0.9157
<i>CDAst8(1)</i>	0.9609	0.9524	0.9588	0.9547
<i>CDAst3(1)</i>	0.9706	0.9663	0.9715	0.9676
<i>CDA(1)</i>	0.9765	0.9743	0.9759	0.9749
<i>CDA(10)</i>	0.9822	0.9748	0.9812	0.9772
<i>STR(10)</i>	0.9834	0.9787	0.9822	0.9803
<i>FT(10)</i>	0.9846	0.9810	0.9836	0.9822
<i>CDA(5) + STR(5)</i>	0.9858	0.9814	0.9849	0.9831
<i>CDA(3) + STR(3) + FT(3)</i>	0.9864	0.9831	0.9856	0.9843
Swin				
<i>None(1)</i>	0.9579	0.9502	0.9578	0.9523
<i>CDA(1)</i>	0.9886	0.9854	0.9886	0.9857
<i>CDA(3) + STR(3) + FT(3)</i>	0.9966	0.9963	0.9963	0.9964

METHOD(n) means the use of a network ensemble; in particular, n stands for the number of neural networks used. Their predictions are fused by sum rule, e.g. *None*(10) means that we combine 10 networks trained using *None*, *CDA*(3) means that we combine 3 networks trained using *CDA*.

$A(n)+B(n)$ means that we combine by sum rule the ensemble $A(n)$ and $B(n)$.

None refers to the use of a neural network without any data augmentation technique.

CDA is the Chromosome Data Augmentation algorithm detailed in Section 2.4; *CDAstx*(1) means that we keep only one image for every x images created by *CDA*, so it is a subset of *CDA*.

STR is the straightening procedure described in Section 2.5, using this approach we apply straightening procedure to both training and test images, moreover, after the straightening procedure we apply *CDA* to the training set; while *FT* refers to the use of the Feature Transform-based techniques presented in Section 2.6, we apply this method to one image out of every three images in the training set, adding the new images to the original training set, then we apply *CDA* to the expanded training set.

Because of the computation time for SWIN, we run only a subset of the tests launched for ResNet-50. From the results reported in Table 3.1 we can get the following conclusions:

- *None*(1) achieves significantly lower performance than the network trained using an expanded training set. *None*(10) outperforms *None*(1), but its performance is not comparable to that achieved by *CDA*(10). *CDA*(1) outperforms both *CDAst8*(1) and *CDAst3*(1), it is clear that in this application data augmentation is a very important step.
- *CDA*(10) clearly outperforms *CDA*(1).
- The best performance, considering a single data augmentation approach, is obtained by *STR*(10).
- The CNN ensemble trained with different augmentation methods can outperform each of its components, e.g., *CDA*(3) + *STR*(3) + *FT*(3) outperforms *STR*(10).
- The best result among CNN methods is obtained by *CDA*(3) + *STR*(3) + *FT*(3), which achieved an accuracy of 98.56%.
- SWIN performs significantly better than ResNet-50, confirming the same conclusions: the ensemble performs significantly better than the stand-alone network; furthermore, we calculated the area under the ROC curve of the SWIN ensemble, obtaining an excellent 0.9999.

We added the learning curve (see Figure 14) of the baseline network (i.e., without data augmentation), which obtains zero loss error, so it perfectly classifies the training after a few training epochs; clearly, the CNN is likely to overfit the training data, so it is important to increase the size of the training ensemble to reduce the risk of overfitting.

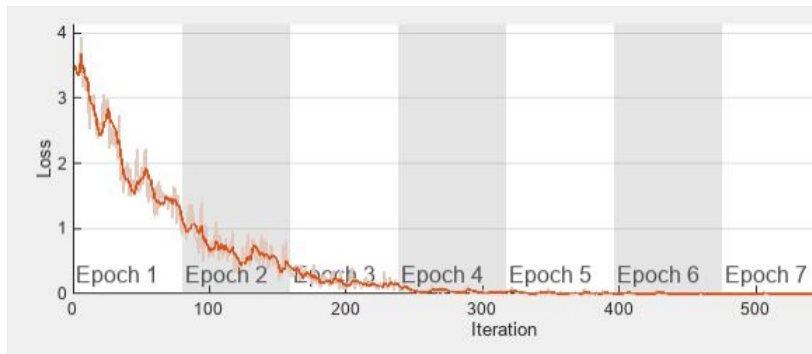


Figure 14: learning curve of a stand-alone ResNet-50.

The primary drawback of the suggested ensemble is the longer computation time it requires. While the preprocessing methods are not computationally intensive, the main computational demand lies in performing inferences with the set of CNNs. Nonetheless, using a Titan RTX 24 GB, it is possible to classify using ResNet-50 a batch of one thousand images in just 1.945 seconds (therefore, an ensemble of 10 ResNet-50 networks can classify 100 images in ~ 2 seconds). The data augmentation approaches and the training of ResNet-50 are implemented in Matlab (we have used the 2023a version); for SWIN we used the Python Timm library.

Finally, we should stress the main cons of using this data set as a benchmark, we have no information about which karyotype a given image belongs to, so chromosome images are randomly split in 5-fold cross-validation; a better testing protocol should be splitting whole karyotypes between training and testing (i.e., a whole karyotype with all images in the training or test set).

Chapter 4

Conclusions

In this study, we focused on automating the karyotyping procedure, specifically the classification of human chromosome images. We provided an overview of the context and an in-depth analysis of three image processing and data augmentation techniques that we adopted to achieve state-of-the-art performance:

- A data augmentation algorithm called *CDA* was used to generate additional samples from the original dataset. This algorithm introduces different spatial orientations to the chromosomes, effectively diversifying the training data.
- We implemented a straightening procedure that utilizes projection vectors to straighten the chromosomes. This step removes curves from the subjects, allowing the neural network to learn other important features.
- In the end, we employed three feature transform-based techniques to create more images and alter their appearance through manipulations such as blur and contrast adjustments. These techniques contributed to further enhancing the diversity and variability of the training data.

To evaluate the effectiveness of our methods, we conducted experiments on a dataset comprising 2986 human chromosome images. The dataset was used to create five folds for training and testing purposes. Furthermore, we partially used two additional datasets to evaluate the effectiveness of the straightening procedure.

For the classification, we utilized a ResNet-50/SWIN neural network combined with an ensemble approach. This method allows for high accuracy and robust predictions, resulting in state-of-the-art performance.

As future work, we plan to conduct tests on other datasets. The overall performance could be improved by processing the unstraightened curved chromosomes; this can be achieved by finding new tuning parameters or employing different algorithms. Furthermore, considering alternative neural network architectures or ensembles could further improve performance.

All the code was written in MATLAB or Pytorch, and it is freely available on GitHub at <https://github.com/MattiaDAngelo>, ensuring accessibility for the research community.

References

- [1] Wang, X.; Zheng, B.; Li, S.; Mulvihill, J.J.; Liu, H. A rule-based computer scheme for centromere identification and polarity assignment of metaphase chromosomes. *Computer Methods and Programs in Biomedicine* **2008**, *89*, 33–42. <https://doi.org/https://doi.org/10.1016/j.cmpb.2007.10.013>.
- [2] Tjio, J.H.; Levan, A. The Chromosome Number in Man. *Hereditas* **2010**, *42*, 1 – 6. <https://doi.org/10.1111/j.1601-5223.1956.tb03010.x>.
- [3] Remani Sathyan, R.; Chandrasekhara Menon, G.; S, H.; Thampi, R.; Duraisamy, J.H. Traditional and deep-based techniques for end-to-end automated karyotyping: A review. *Expert Systems* **2022**, *39*, e12799. <https://doi.org/https://doi.org/10.1111/exsy.12799>.
- [4] Agam, G.; Dinstein, I. Geometric separation of partially overlapping nonrigid objects applied to automatic chromosome classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1997**, *19*, 1212–1222. <https://doi.org/10.1109/34.632981>.
- [5] Errington, P.A.; Graham, J. Application of artificial neural networks to chromosome classification. *Cytometry* **1993**, *14*, 627–639. <https://doi.org/https://doi.org/10.1002/cyto.990140607>.
- [6] Zhang, W.; Song, S.; Bai, T.; Zhao, Y.; Ma, F.; Su, J.; Yu, L. Chromosome Classification with Convolutional Neural Network Based Deep Learning. In Proceedings of the 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2018, pp. 1–5. <https://doi.org/10.1109/CISP-BMEI.2018.8633228>.
- [7] Swati.; Gupta, G.; Yadav, M.; Sharma, M.; Vig, L. Siamese Networks for Chromosome Classification. In Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), 2017, pp. 72–81. <https://doi.org/10.1109/ICCVW.2017.17>.
- [8] Huang, K.; Lin, C.; Huang, R.; Zhao, G.; Yin, A.; Chen, H.; Guo, L.; Shan, C.; Nie, R.; Li, S. A novel chromosome instance segmentation method based on

- geometry and deep learning. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1–8. <https://doi.org/10.1109/IJCNN52387.2021.9533523>.
- [9] Abid, F.; Hamami, L. A survey of neural network based automated systems for human chromosome classification. *Artificial Intelligence Review* **2018**, *49*, 41–56. <https://doi.org/10.1007/s10462-016-9515-5>.
- [10] Anh, L.Q.; Thanh, V.D.; Son, N.H.H.; Phuong, D.T.K.; Anh, L.T.L.; Ram, D.T.; Minh, N.T.B.; Tung, T.H.; Thinh, N.H.; Ha, L.V.; et al. Efficient Type and Polarity Classification of Chromosome Images using CNNs: a Primary Evaluation on Multiple Datasets. In Proceedings of the 2022 IEEE Ninth International Conference on Communications and Electronics (ICCE), 2022, pp. 400–405. <https://doi.org/10.1109/ICCE55644.2022.9852034>.
- [11] Lin, C.; Zhao, G.; Yang, Z.; Yin, A.; Wang, X.; Guo, L.; Chen, H.; Ma, Z.; Zhao, L.; Luo, H.; et al. CIR-Net: Automatic Classification of Human Chromosome Based on Inception-ResNet Architecture. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **2022**, *19*, 1285–1293. <https://doi.org/10.1109/TCBB.2020.3003445>.
- [12] Javan Roshtkhari, M.; Setarehdan, K. A novel algorithm for straightening highly curved images of human chromosome. *Pattern Recognition Letters* **2008**, *29*, 1208–1217. <https://doi.org/10.1016/j.patrec.2008.01.029>.
- [13] Sharma, M.; Saha, O.; Sriraman, A.; Hebbalaguppe, R.; Vig, L.; Karande, S. Crowdsourcing for Chromosome Segmentation and Deep Classification. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp. 786–793. <https://doi.org/10.1109/CVPRW.2017.109>.
- [14] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. <https://doi.org/10.1109/CVPR.2016.90>.
- [15] MathWorks. ResNet-50 convolutional neural network - MATLAB resnet50 - MathWorks. <https://it.mathworks.com/help/deeplearning/ref/resnet50.html?lang=en>, n.d. Accessed: April 27, 2023.
- [16] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need, 2017, [arXiv:cs.CL/1706.03762].
- [17] Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. Transformers: State-of-the-Art Natural

- Language Processing. In Proceedings of the Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations; Association for Computational Linguistics: Online, 2020; pp. 38–45. <https://doi.org/10.18653/v1/2020.emnlp-demos.6>.
- [18] Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2021, [arXiv:cs.CV/2010.11929].
- [19] Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, 2021, [arXiv:cs.CV/2103.14030].
- [20] Grisan, E.; Poletti, E.; Ruggeri, A. Automatic Segmentation and Disentangling of Chromosomes in Q-Band Prometaphase Images. *IEEE Transactions on Information Technology in Biomedicine* **2009**, *13*, 575–581. <https://doi.org/10.1109/TITB.2009.2014464>.
- [21] Poletti, E.; Grisan, E.; Ruggeri, A. Automatic classification of chromosomes in Q-band images. In Proceedings of the 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2008, pp. 1911–1914. <https://doi.org/10.1109/IEMBS.2008.4649560>.
- [22] Ritter, G.; Gao, L. Automatic segmentation of metaphase cells based on global context and variant analysis. *Pattern Recognition* **2008**, *41*, 38–55. <https://doi.org/https://doi.org/10.1016/j.patcog.2007.05.013>.
- [23] Lin, C.; Yin, A.; Wu, Q.; Chen, H.; Guo, L.; Zhao, G.; Fan, X.; Luo, H.; Tang, H. Chromosome Cluster Identification Framework Based on Geometric Features and Machine Learning Algorithms. In Proceedings of the 2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2020, pp. 2357–2363. <https://doi.org/10.1109/BIBM49941.2020.9313369>.
- [24] Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data* **2019**, *6*. <https://doi.org/10.1186/s40537-019-0197-0>.
- [25] Moradi, M.; Setarehdan, S.; Ghaffari, S. Automatic locating the centromere on human chromosome pictures. In Proceedings of the 16th IEEE Symposium Computer-Based Medical Systems, 2003. Proceedings., 2003, pp. 56–61. <https://doi.org/10.1109/CBMS.2003.1212767>.