DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2020

Chromosome Classification using Machine Learning to enable Genetic Disorder Diagnostics

An application for Arkus.ai

MOHAMED HASSAN HAMZA

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Chromosome Classification using Machine Learning to enable Genetic Disorder Diagnostics

MOHAMED HASSAN HAMZA

Master in Computer Science
Date: October 9, 2020
Supervisor: Ying Liu
Examiner: Amir H. Payberah
School of Electrical Engineering and Computer Science
Host company: Arkus.AI, Sweden

Abstract

Chromosome enumeration is an indispensable but monotonous procedure in chromosome karyotyping analysis which is an essential process for genetic disorder diagnostics. To automate the enumeration process, we have developed a chromosome enumeration framework based on the region based object detection scheme using supervised machine learning. We apply several data augmentation techniques on our dataset, i.e. Karyogram images. Then, we developed and trained four different classifiers which are SSD, Faster RCNN, RFCN, and YOLO v5 to predict the class of each chromosome. The results showed that YOLO v5 outperformed SSD, Faster RCNN, and RFCN. Finally, using our model that can predict the class of each chromosome, we can automatically generate Karyograms using the original chromosome images.

Sammanfattning

Kromosomuppräkning är en väsentlig men monoton procedur i kromosomkaryotypanalys, vilket är en viktig process för genetisk störningsdiagnostik. För att automatisera uppräkningsprocessen har vi utvecklat ett ramverk för kromosomuppräkning baserat på det regionbaserade objektdetekteringsschemat med hjälp av övervakad maskininlärning. Vi tillämpar flera dataförstärkningstekniker på vår dataset, dvs. karyogram-bilder. Sedan utvecklade vi och utbildade fyra olika klassificeringsapparater som är SSD, Faster RCNN, RFCN och YOLO v5 för att förutsäga klassen för varje kromosom. Resultaten visade att YOLO v5 överträffade SSD, Faster RCNN och RFCN. Slutligen, med hjälp av vår modell som kan förutsäga klassen för varje kromosom, kan vi automatiskt generera karyogram med originalkromosombilder.

Contents

List of Figures

Chapter 1

Introduction

Automation of genetic diagnostic has been a major research topic in the last few years. The process of diagnosing genes and chromosomes is very complex and requires years of training for doctors to be able to diagnose with a small percentage of error. In chromosome abnormality diagnostics, the unbalance of normal and abnormal data makes it more challenging for trainees to learn how abnormal chromosomes look like because the majority of chromosomes are normal. As a result, doctors need more data and more years of training to gain expertise in this field.

Chromosome abnormality refers to the anomaly, disorder, mutation, a missing, extra, or irregular sections of chromosomal DNA. According to studies done by Gert de Graaf [1], the number of Down-Syndrome (DS) is 12.6 per 10,000 with an average of 5300 births annually in the years (2006-2010) in the United States. By 2007, the estimated number of DS births decreased by 30% due to the DS-related implementation of the medical project in the United States, see figure 1.1. Hence, Prenatal screening and diagnosis of chromosomal diseases gain massive importance as it significantly reduces the incidence of birth defects.

	Estimates of nonselective live birth prevalence of DS (per 10,000 live births)	Estimates of live birth prevalence of DS (per 10,000 live births)	Estimated reduction percentage
Asians/Pacific Islanders	25.0 (19.2–30.8)	10.0 (9.4–10.6)	60.1% (50.0–70.2)
Non-Hispanic whites	26.6 (24.6–28.7)	12.3 (12.2–12.4)	53.8% (50.2–57.5)
Non-Hispanic blacks/Africans	22.4 (17.3–27.4)	15.8 (15.1–16.5)	29.4% (12.1–46.7)
Hispanics	15.7 (12.4–19.1)	13.7 (13.2–14.2)	12.7% (0–32.8)

95% CI in parentheses.

DS, Down syndrome; MA, Massachusetts.

Figure 1.1: Estimates of non-selective and live birth prevalence of Down Syndrome and reduction percentage in Massachusetts for 2006–2010 [1].

A karyotype is the number, size, and shape of chromosomes in an organism. To determine the karyotype of an organism, scientists must follow these steps:

- Collect a cell from an individual
- Induce the cell to divide
- Stop cell division in metaphase when chromosomes are easiest to see
- Stain the chromosomes to make them visible
- View the cell under a microscope
- Another entry in the list

A regular Human has a total of 46 chromosomes (23 from each parent), which you can see in the *Karyograms*. In a Karyogram, homologous chromosomes, or pairs of chromosomes that are the same size and shape, are grouped together. Humans have 22 pairs called *Autosomes*. Autosomes are the chromosomes that contain genes for determining all human characteristics except the determination of human sex. Autosomes are numbered from class 1 through class 22, and we have two of each class because one came from your mom and one came from your dad. Class 1 chromosome is the largest, and class 22 chromosome is the smallest. Autosomes only account for 44 of your total chromosomes, the remaining two chromosomes in humans are called sex chromosomes or chromosomes X and Y because they are the ones that determine whether you are male or female. They are placed after autosomes in a Karyogram as shown in figure 1.2. If someone has two X chromosomes, it is a female. If someone has an X and a Y chromosome, it is a male.

1.1 Motivation

Karyotyping is a laboratory procedure that allows doctors to examine people's set of chromosomes. A Karyotype refers to the actual collection of chromosomes that are being examined by doctors. Examining chromosomes through karyotyping allows doctors to determine any abnormalities or structural problems within the set of chromosomes.

Chromosomes are in every cell of your body. They contain the genes that are inherited from our parents. Genes determine the way every human develops and they are composed of Deoxyribose Nucleic Acid (*DNA*). Cells need to

pass on a complete set of genetic instructions to each newly formed cell during cell division and cells are lined up in pairs. When there is no cell division, the chromosomes are arranged in an unorganized way. A karyotype test investigates these dividing cells. Each pair of chromosomes is arranged by its size and appearance. Helping doctors to easily determine if any chromosome is missing or damaged.
The process of Karyotyping is typically done by a Cytogeneticist and it takes long time and expertise to do it accurately with a low error rate. Our main motive is to develop a tool that helps to automate the process of Karyotyping, helping doctors and patients who wait in long queues to have their genetic test done. First, we create a machine learning model that creates bounding boxes around each chromosome, then we generate a new image and compare the size of each chromosome and align them in a way similar to the Karyogram arranged from largest to smallest. Finally, we train our machine learning model for abnormality detection using the newly generated images.

1.2 Problem

Chromosome genetic disorder diagnostics is currently done by Cytogeneticists. First, they look into the cells using a microscope and take around 96 images of the chromosomes from different angles, then they select 5 images with the fewest overlapping. Then using these images, they create another image called Karyogram using the selected chromosomes, an example of a Karyogram is shown in Figure 1.2. A Karyogram is a way used to depict chromosomes, chromosomes are organised in the image in a way that makes them easy to visualize. They are arranged into homologous pairs where each is arranged in order from largest to smallest.

The research question that this thesis tries to solve can be phrased as: *"Can machine learning and image processing approaches achieve good accuracy in generating Karyograms?"*

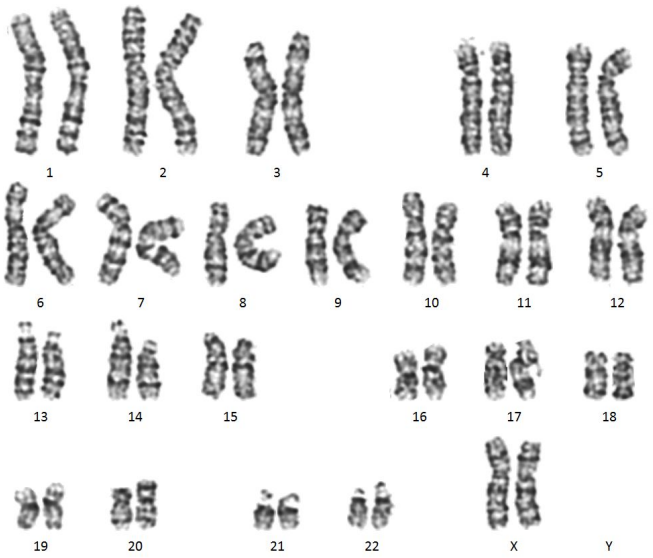

Figure 1.2: An example of a Karyogram

1.3 Goals

The goal of this project is to create an application that automates chromosome abnormality diagnostics with image processing and machine learning techniques using the original chromosome images. Our research has shown that using only the unorganized chromosome set shown in figure 1.3 will result in dramatically low predictions for our machine learning model due to the high complexity even if we have hundreds of thousands of images.

Figure 1.3: Original Chromosome image

1.4 Thesis contributions

The contribution of this thesis includes implementing several machine learning classifiers for the task of chromosome classification and detection. We implemented the machine learning models, faster *RCNN Inception v2*, *RFCN Resnet101*, *SSD Mobilenet v2* and *YOLO v5* for our image processing classifiers. Then we did hyperparameter tuning to optimize our models to achieve the best possible recall and precision results.

1.5 Ethics and sustainability

This thesis addresses problem that could have a huge societal impact. Diagnosing humans early for any genetic abnormality will not only free up resources in the resource-starved healthcare system around the world but will also mitigate the adverse outcomes of genetic abnormalities on people. Thereby it could aid physicians or health workers to concentrate the fewer health resources towards the subjects who are actually in need of it and in a timely manner. This project has been proposed by the host company for their internal research and appropriate *GDPR* regulations were met before carrying out work on this dataset. The dataset is primarily medical in nature where the predictor variables are already chosen and anonymized in an appropriate manner such that leaving no grounds to link the data to any of the patients from whom it has been collected.

1.6 Outline

Chapter 2 presents the theoretical background of the concepts that are employed in developing our machine learning classifiers and their performance metrics. Chapter 3 dives into the related research work and papers that have been published relating to our project. Chapter 4 presents the research methodologies that are incorporated to study our problem statement. Chapter 5 presents and discusses the results of our project. Chapter 6 concludes this thesis work and provides insight for future work.

Chapter 2

Background

2.1 Machine Learning in image processing

There are 3 key learning paradigms in image processing tasks, supervised, unsupervised and reinforcement learning as shown in Figure 2.1. Supervised learning is learning using pre-labelled data which are used as inputs to the machine learning model. For each training example, there will be a set of values as inputs and one or more output values associated. This form of training is used to reduce the model's overall classification error, through the correct calculation of the output value of the training example [2]. On the other hand, Unsupervised learning's training set does not include any labels. Success is measured by whether the neural network is able to reduce or increase an associated cost function. Although, most image-focused pattern-recognition tasks often depend on classification with supervised learning.

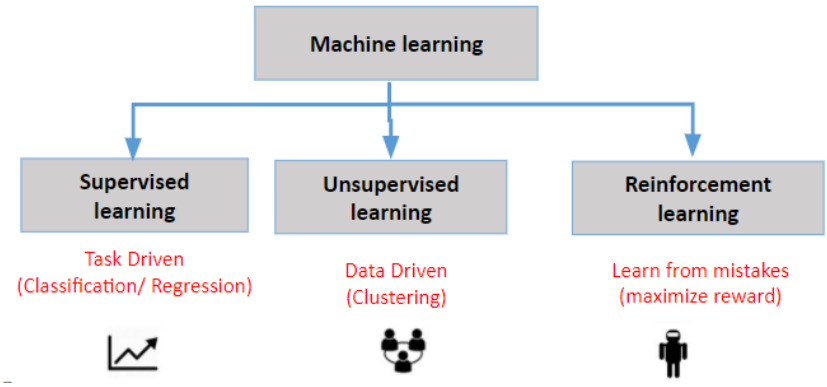

Figure 2.1: Describing different machine learning techniques with some use-case examples

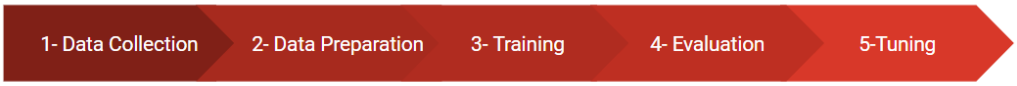

Figure 2.2: Machine learning process

As shown in Figure 2.2, the process of machine learning can be defined in the following steps:

1. **Data Collection** : Collect the data for training the model.
2. **Data Preparation**: Extracting important features, format the data into the optimal format, and performing dimensionality reduction.
3. **Training**: The model is selected and trained using the data that we prepared in the previous step.
4. **Evaluation**: Gaining insights from the model's results, and evaluating how well the model performs.
5. **Tuning**: In order to maximize the model performance we perform fine tuning.
6. **Repeat from Step 3.**

2.1.1 Supervised machine learning

Supervised Machine Learning (*SML*) requires human guidance for a specific training dataset of labelled examples data, for instance, when SML algorithm is used, the algorithm is supported by a dataset containing classified data (Historical) with known inputs and outputs aiming to train the model on predicting the output through identifying the input itself, see Figure 2.3. A well known SML application is image recognition, for instance, to identify whether there is an apple in a picture or not, SML algorithm is used for checking whether there is an apple in the picture or not through historical labelled data of apple pictures, in the previous example, the dataset will be thousands of apple pictures with a variety of types, shapes, and colours, the algorithm will learn how to identify whether that input is an apple or not later on.

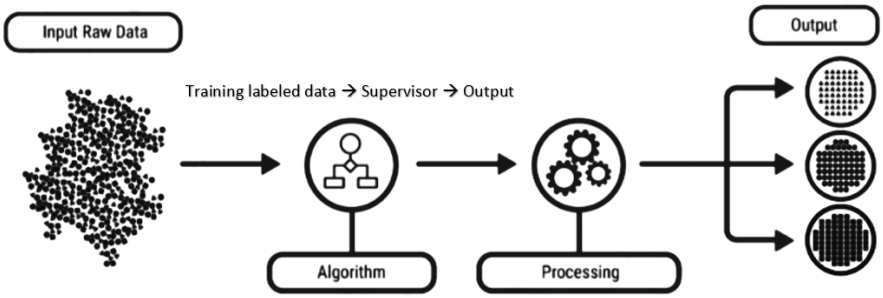

Figure 2.3: Structure of supervised machine learning

A good supervised machine learning model must be able to adapt to new inputs and make predictions. During model training, we need to maximise generalisation, thus, the supervised model defines a 'general' underlying relationship for the data. If we overtrain the model, this will cause overfitting to the inputs used and the model will not be able to adapt to new or unseen inputs. On the other hand, we must be aware of a major side-effect in supervised learning which is that the our supervision to the model introduces bias to the model prediction. The model can only learn what it was provided, so it is crucial to show it reliable, unbiased examples. Moreover, supervised learning usually requires large amounts of labelled data in order to have accurate predictions. However, Acquiring enough reliably labelled data is often the most difficult and expensive part of using supervised learning.

2.1.2 Classification

Classification is used to group similar data points into different sections in order to classify them. For this task, we use machine learning to determine the rules that explain how to identify different data points. Rules focus on using data and answers to discover rules that linearly separate data points. There are many ways to discover the rules. The most common rule is Linear separability which is a key concept in machine learning where the main task of it is to find the best way to separate data-points/classes with a line which is known as the decision boundaries. The entire area that is chosen to define a class is known as the decision surface. As shown in Figure 2.4, If a data point falls within the boundary of the decision surface it will be assigned a certain class [3].

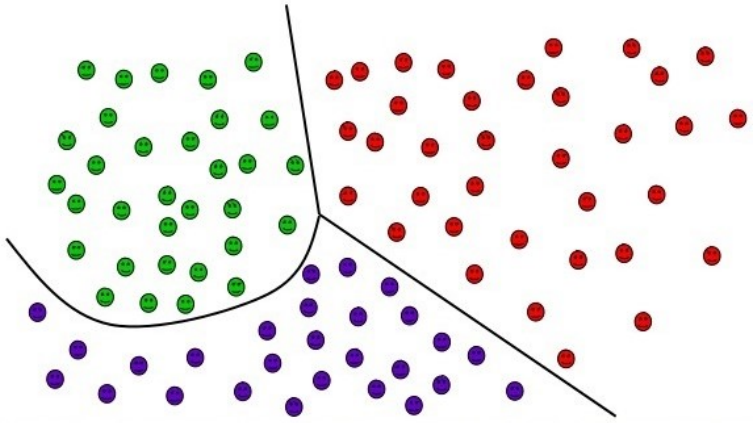

Figure 2.4: Multi-class Classification example

2.1.3 Object Detection

Object recognition is a technique in computer vision that is used to identify different objects in images or videos. Deep learning and machine learning algorithms are the main techniques for performing object detection. The goal for these techniques is to show the computer how to identify objects the same way humans do, i.e. to have an understanding of what an image contains. Driverless cars are examples for systems that uses object detection, machine learning teaches the the car's system to recognize a stop sign and distinguish a pedestrian from other objects in the street. Moreover, It is useful in a variety of applications such as medical diagnostics, robotics, and face recognition.

Most modern object detection applications are using Deep learning algorithms. Deep learning models such as CNNs, are used to learn features in

objects in order to identify that object. For instance, CNNs can learn to differentiate between apples and oranges by analyzing thousands of images and learning the features that make apples and oranges different.

Performing a deep learning based object recognition can be done using 2 different methods:

- Train a model from scratch: In this method, we need to gather a large dataset and design a neural network algorithm that will learn the features and build the model. The results of this method can be outstanding, however, it needs massive amounts of data to achieve good predictions and you need to add weights and layers in the network.

- Using a pretrained deep learning model: This is the most common technique in most applications, and it involves fine-tuning a pretrained model. You start with an existing network, such as Inception V2, and feed the network with new data containing classes unknown to the network. This method much faster and because the model has already been pre-trained on thousands of images.

2.1.4 Convolutional Neural Networks

Convolutional Neural Networks (*CNN*) are models that simulate how the human brain processes information. This is typically done by developing a large number of interconnected processing units (nodes) that represents an abstract version of neurons.

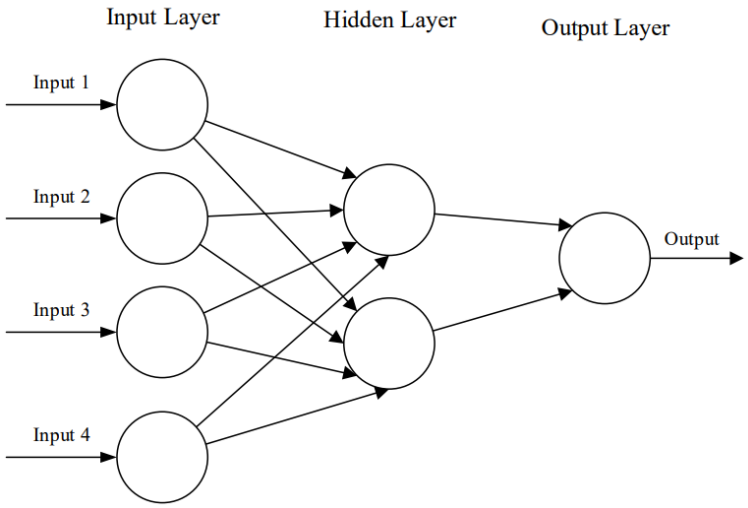

Figure 2.5: Example of a neural network [2].

As shown in Figure 2.5, the most basic neural network is arranged in 3 layers, an input layer, one or multiple hidden layers; and an output layer, with a unit representing the target field. Each unit is connected with different connection weights. The first layer is storing the input data, while values are propagated from each neuron to all neurons in the next layer. Finally, a result is delivered from the output layer [4].

The CNN architecture shows the best performance by processing multimedia data formats such as image and videos. The visual cortex is a complex arrangement of the cell which are sensitive to a small region of the visual field called a receptive field. Generally, ANN uses a matrix multiplication which is replaced in CNN by using Convolution layers. Convolution layers are used to reduce the complexity of the network by reducing the number of weights. Consequently, there is no need of separate feature extraction algorithm, which is popular in standard learning algorithms.

Similarly to a neural network, CNN consists of input, output and hidden layers in between. These hidden layers performs feature detection and it performs three operations on data; Convolution, Pooling and Rectifier Linear Unit (*ReLu*). First, The convolution layer is used to learn certain features of an image, by inserting the input image into a set of convolutional filters. Each filter learns different features from the image. Secondly, the pooling layer helps reducing the number of irrelevant parameters by simplifying the output using nonlinear downsampling method [5]. Thirdly, ReLu improves the efficiency and learning speed of the network by mapping negative values to zeros in order

to maintain only positive values. These three different operations are applied repeatedly to tens or thousands of layers to be able to detect different levels of features. Before the last layer there is a fully connected layer that outputs an N dimension vector, where N is the number of output classes the network is able to predict. Finally, The last layer is a *softmax* layer used to provide the classification output. The components of the CNN is depicted in the following figure 2.6

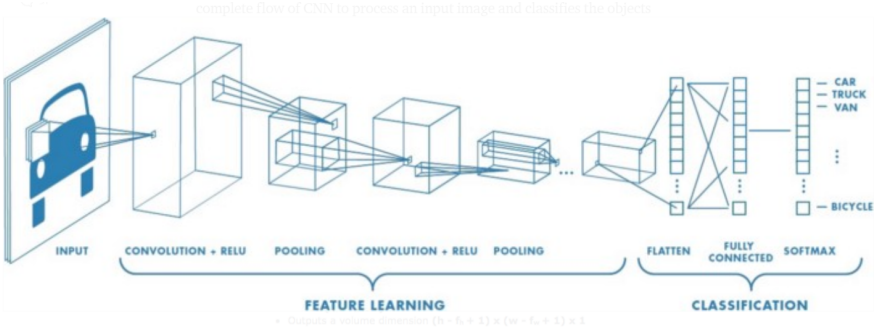

Figure 2.6: The architecture of a Convolution neural network [5].

2.1.5 Faster RCNN

The current advancement of object detection is driven by the region-based convolutional neural networks (*RCNN*) [6]. During the early development RCNN techniques in [7], the computational costs were very high. However, their cost has been reduced by using sharing convolutions across proposals. The latest advancement of RCNN is faster RCNN, using very deep networks, faster RCNN achieves real-time speeds when ignoring the time spent on region proposals. Currently, proposals are considered the computational bottleneck in object detection systems.

As shown in Figure 2.7, Faster RCNN is based on 2 models, Region Proposal Network *RPN*) and Fast RCNN. RPN is based on the fully convolutional networks [8], it can predict the target area frame and confidence score (the probability of a correct prediction) at each region of the input picture. RPN is used to generate high quality bounding boxes around target objects, it shares the convolution feature of the whole graph with the detection network and solves the speed problem of the original Selective Search [9], which significantly improves the performance of object detection. Furthermore, Faster RCNN uses Fast-RCNN for classification and object detection, this method solves the problems of slow speed of RCNN detection and large memory con-

sumption. Fast RCNN uses the quality region proposal which is provided by RPN for target recognition which greatly improved the speed and accuracy of object detection.

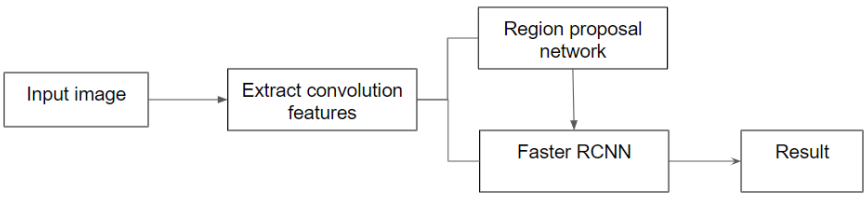

Figure 2.7: Faster RCNN structure

2.1.6 YOLO CNN

You Only Look Once (*YOLO*) is a real-time object detection system which is famous for its simplicity and speed. As shown in figure 2.8, YOLO applies a single neural network to the whole image, it divides the image into regions and predicts multiple bounding boxes and class probabilities(confidence score) for those boxes. These bounding boxes are weighted by the predicted probabilities. YOLO trains on full images and directly optimizes detection performance. Thus, its predictions are informed by the global context in the image. Unlike sliding window and region proposal-based techniques, as YOLO processes the entire image during training and test time, it encodes contextual information implicitly about classes as well as their appearance. Fast RCNN [10], often mistakes background patches in an image for objects because it does not see the larger context. YOLO makes less than half the number of background errors compared to Fast RCNN. Using a single neural network makes YOLO around 100x faster than Fast RCNN and 1000x faster than RCNN as these models require thousands of networks for a single image.

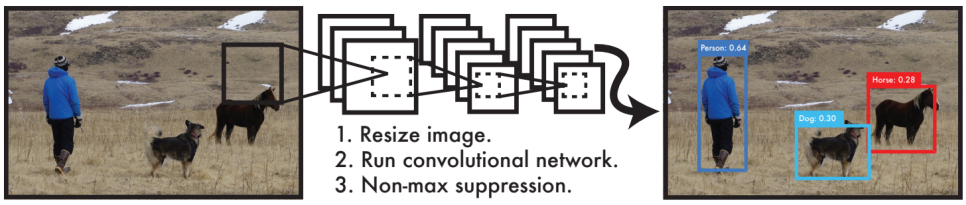

Figure 2.8: The YOLO Detection System. Processing images with YOLO is simple and straightforward. First, the YOLO system resizes of the input image to 448 x 448 pixels, then it runs a single convolutional network on the image, finally, it thresholds the resulting detections by the model's confidence [11].

YOLO's speed could reach 155 frames per second, achieving double the mAP of other real-time detectors like Fast RCNN. YOLO's base network runs at 45 frames per second without batch processing on a Titan X GPU and some faster versions runs at more than 150 fps. Consequently, we can do video stream processing in real-time with less than 25 milliseconds of latency. Moreover, YOLO achieves more than twice the mean average precision of other real-time systems.

YOLO learns the general representations of objects. YOLO outperforms top detection methods like Deformable Part Models *DPM* and RCNN by a wide margin When trained on natural images and tested on artwork. Since YOLO can be used generally in many use cases. It is less likely to break down when applied to unexpected inputs or new domains. On the other hand, YOLO makes more localization errors than other state-of-the-art systems but is less likely to predict false positives on the background. Furthermore, it lags behind in terms of accuracy compared to state-of-the-art detection systems. YOLO can quickly identify objects in images, although, it struggles to localize small objects precisely.

As shown in figure 2.9, the detection network consists of 24 convolutional layers followed by 2 fully connected layers. Alternating 1 x 1 convolutional layers to reduce features space from preceding layers. The model is pretrained on the *ImageNet* classification task at half the resolution (224 x 224 input image) and then double the resolution for detection.

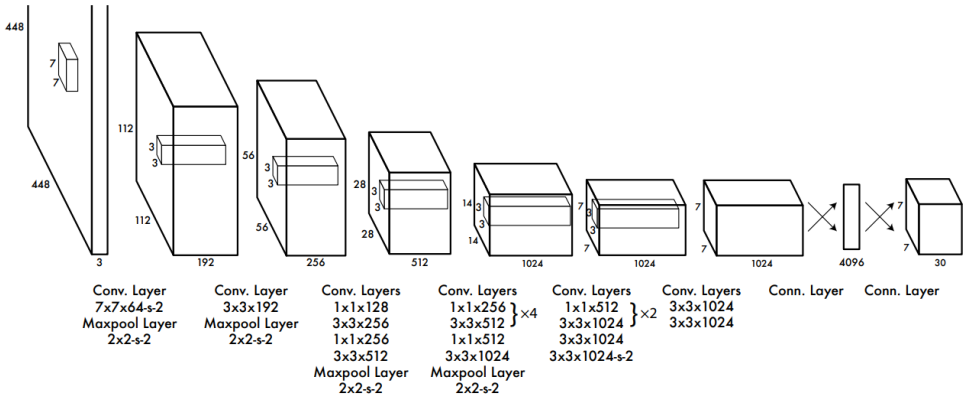

Figure 2.9: YOLO's Architecture [11].

2.1.7 Microsoft COCO: Common Objects in Context

Microsoft COCO dataset was created to advance state-of-the-art object detection by placing the question of object detection in the context of the broader question of understanding scenes. Microsoft achieved this by gathering images of complex scenes containing common objects where the images were taken in their natural contexts [12].

2.2 Performance measures for the machine learning model

There are various metrics that we can use to evaluate the performance of machine learning algorithms. These metrics must be identified for evaluating ML performance for various reasons. Firstly, how the performance of ML algorithms is measured and compared will be dependent entirely on the chosen metric. Secondly, how you weight the importance of various characteristics in the result will be influenced completely by the metric you choose. In this section, we discuss the chosen metrics for evaluating our ML models.

2.2.1 Precision and Recall

Precision, also known as positive predicted value is defined as the fraction of relevant instances among all the retrieved instances. In other words, precision describes the percentage of the results that are relevant. Recall, which is in the medical diagnostics terms widely known as sensitivity, is the percentage of

total relevant results that are labelled correctly by our model. Both precision and recall are a measurement of the relevance of the results. In a classification task, which is our main task in this project, the precision for a class is defined as the number of objects of response variable correctly labelled as belonging to the positive class divided by the total number of objects of response variable labelled as part of the positive class by our model. In the same context, Recall is defined as the number of objects of response variable labelled correctly as belonging to the positive class divided by the total number of objects of the response variable that in original belong to the positive class including objects that were not labelled as belonging to the positive class by the model.
A perfect precision score of 1.0 for a class, let's say class 1 chromosomes, would mean that every instance labelled as belonging to class 1 indeed is a class 1 object, but precision doesn't highlight the number of instances from class 1 objects that were not labelled correctly as. A perfect recall of 1.0 signifies that every chromosome of class 1 was labelled as belonging to class 1, but recall doesn't highlight how many class 1 objects were incorrectly labelled as any other class. This also highlights an inverse relationship between Precision and Recall, where it is possible to increase either one of the two quantities often at the expense of others.
Mathematically, precision and recall are described in the following equations:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.1}$$

And Recall is defined as

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.2}$$

Figure 2.10 explains how recall and precision in a simplified way.

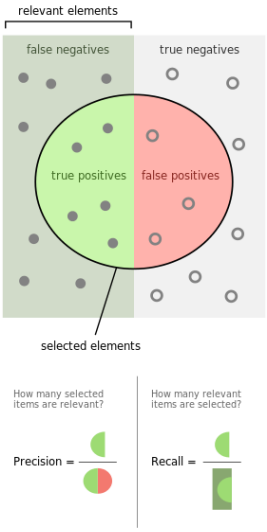

Figure 2.10: Difference between Precision and Recall[13].

.

2.2.2 F1 Score

In many image classification and object detection tasks, increasing precision will lead to a decrease in recall or vice-versa. Hence, using these two quantities to measure the performance of our model could be tricky. However, there is a simpler metric, F1 score, which takes into consideration both precision and recall. Hence, our goal is to maximize this number to make our model better. F1 score can be defined as the weighted average of precision and recall. Both the precision and recall have an equal contribution to the calculation of the F1 score. F1 score assumes values between 0, the worst score, and 1, the best-score.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.3}$$

Chapter 3

Related Work

The paper Abnormality Detection and localization in Chest X-Rays (CXR) using deep neural networks proposes a scheme for automated abnormality detecting with deep neural networks on the Indiana chest X-Ray dataset, JSRT dataset and Shenzhen dataset [14]. The solution used heat maps obtained from occlusion sensitivity as a measure of localization in the CXRs. The research conducted the following findings:

- Multiple random train/test data split achieve good accuracy results when the number of training examples is low.
- Shallow features or earlier layers have better performance than deep features for classification accuracy.
- Same Deep Convolutional Networks (DCN) architecture doesn't perform well across all abnormalities.
- Ensemble of DCN models has better performance than single models. Although, mixing DCN and rule-based ensemble model decreases accuracy.
- The application of the methods in this paper achieved the highest accuracy for tuberculosis detection.
- The results show a 17 percent improvement in accuracy over rule-based methods for Cardiomegaly detection using DCN.

Another paper by Chouhan et. al. [15] discuss an approach bases on transfer learning for pneumonia detection in chest X-ray images. Pneumonia is one

of the top diseases that cause most of the deaths around the world. Several factors such as Virus, Bacteria and Fungi can cause Pneumonia. Although, judging Pneumonia is extremely difficult just by looking at the chest X-rays. Hence, researchers are trying to solve this problem to make it easier for doctors the decision of whether this chest has pneumonia or not. In this paper, researchers suggested a deep learning framework using transfer learning [16] for detecting pneumonia. Features are extracted using different pre-trained neural networks on ImageNet, then these models are fed to a classifier for prediction. The paper proposed an ensemble model that gathers outputs from the pre-trained models which outperformed individual models, then combines them. This method reached the state-of-the-art performance in pneumonia detection, reaching an accuracy of 96% and recall of 99.62% on data from the Guangzhou Women and Children's Medical Center dataset. First, they do chest X-ray image preprocessing, followed by data augmentation, transfer learning using Inception v3, AlexNet, DenseNet121, GoogLeNet, and Resnet18 neural networks. Finally, they did feature extraction and ensemble classification.

Tuberculosis (TB) is a common lung disease that targets millions of people every year of all different age group around the world. Rajaraman [17] discusses a scheme based on deep convolutional neural networks to automatically detect TB using chest CXR images. Figure 3.1 shows CXR images of normal lungs, lungs with pleural effusion, and cavitary lung lesion.

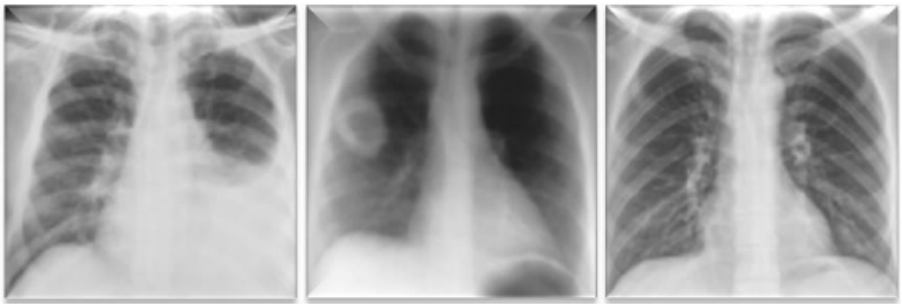

Figure 3.1: examples of CXR images showing pulmonary abnormalities (left: pleural effusion, middle: cavitary lung lesion, and right: normal lung image) [17].

The research evaluated the performance of a stacked ensemble which combined classifiers using hand-engineered feature extraction methods in addition to those extracted from the pre-trained CNNs using two proposals to improve the accuracy of TB detection in CXR images. The first proposal uses the features GIST, HOG, and SURF extracted from CXR images and trained an SVM

classifier to predict whether the chest is normal or abnormal.In the second proposal, the authors used four CNN pre-trained models, AlexNet [18], VGG-16 [19], ResNet-50 [20], and GoogLeNet [21] for feature extraction from CXT images and trained a Support Vector Machine (SVM) classifier to detect abnormal images that show TB manifestation.

Automated classification of chromosomes was discussed in [22], the paper proposes a new karyotyping scheme using a two-layer classification platform, this reduced complexity of automated karyotyping and improved its performance. Furthermore, the paper suggested a mechanism that selects the most significant and effective feature sets and optimizing classifiers in an adaptive manner for the different groups of chromosomes with similar image characteristics. Using 6900 chromosome images, a genetic algorithm was applied to the data to optimize the topology of multi-feature Artificial Neural Networks (ANN). The first layer of the scheme has a single ANN that classifies the 24 chromosomes into seven classes, each group represents a group of classes. Groups A,B,C,D,E,F,G represents classes 1-3, 4-5,6-12-X, 13-15, 16-18, 19-20, 21-22-Y respectively. Table 3.1 shows a detailed explanation of the Denver group chromosome classification. The results have shown an accuracy of 91.1% of all classes. The second layer consists of seven ANNs that are optimized for seven classes for identifying individual chromosomes. The scheme was evaluated using a 'train–test–validate" method.

Note: Metacentric centromere is located in the middle of a chromosome, submetacentric lies between middle and end of a chromosome, and acrocentric centromere is close to the tip of a chromosome.

Chromosome Class	*Size*	*Relative Position of Centromere*
Group $A(1-3)$	Large	Metacentric
Group $B(4-5)$	Large	Submetacentric
Group $C(6-12,X)$	Medium	Submetacentric
Group $D(13-15)$	Medium	Acrocentric
Group $E(16-18)$	Relatively short	Submetacentric
Group $F(19-20)$	Short	Metacentric or Submetacentric
Group $G(21-22,Y)$	Short	Acrocentric

Table 3.1: Chromosomes classification based on Denver group classification [22]

The paper proposes a technique for feature computation to extract features

from the chromosome images. They first used a thinning algorithm to detect the medial axis where some pixels are missing near both ends of the chromosome and some redundant pixels are added around the middle part of the chromosome. Then an interpolation algorithm is used for connecting each selected fifth pixel and generate a new smoothed medial axis than can remove the redundant pixels. Figure 3.2 shows a flow diagram of the automated classification of chromosomes.

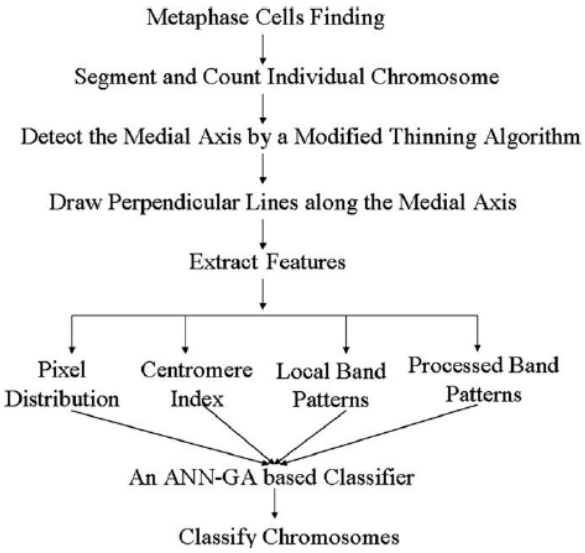

Figure 3.2: A flow diagram of automated classification of chromosomes [22]

For retrieving the absent pixels near both ends of the axis, the algorithm looks for the tip pixels based on the interpolation of previous slopes of the medial axis. Then the medial axis is connected. The final step checks whether the ending pixels reach the exterior contour of the chromosome; if this condition happens, the process is completed and the optimal medial axis is detected. Else, the algorithm retraces two ending pixels iteratively at the medial axis till they reach the exterior contour of the chromosome. A visualization of medial axis detection is shown in figure 3.3.

Figure 3.3: Several medial axis detection results of chromosome 1 with different morphologies obtained by a modified thinning algorithm [22].

Chapter 4

Machine learning models for chromosome classification

The purpose of this chapter is to give an overview of the research methods, methodologies, and engineering approaches used in this project.

4.1 Research Methodology

The main task for this project is to automate the process of generating Karyograms by detecting the class of each chromosome in a microscopic image of human chromosomes. In order to analyse human chromosomes for abnormalities, we need to have a Karyogram image which is typically done manually. Moreover, implementing a machine learning classifier using original chromosome images for abnormality detection is too complex and will result in a very low recall, precision and accuracy scores. Our methodology for solving this problem is as follows; First, we implement several data augmentation our karyogram images that we received for one of the labs in China which are 75 original and Karyogram images, an example is shown in figure 4.1. We developed scripts to do several data augmentations such as 90, 180 degrees rotations, random rotations, reflections/mirroring, random change in brightness, noise injections, and random cropping in order to increase the number of images, reduce overfitting of our model and improve our classifier's recall, precision and accuracy.

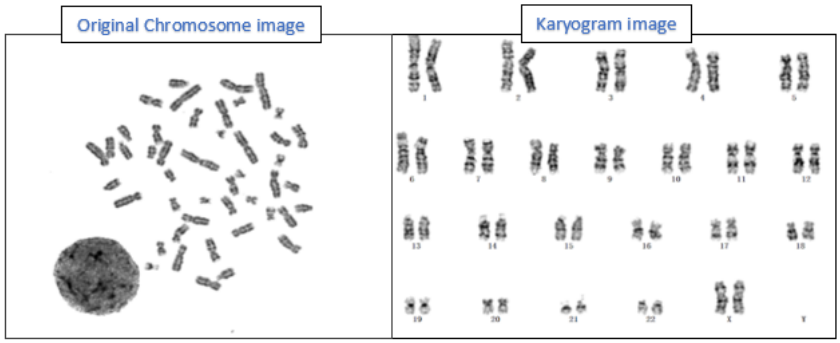

Figure 4.1: Original vs Karyogram images

Secondly, we develop several convolutional neural network classifiers for chromosome detection and classification using the augmented Karyogram images. The models we used for classification are RFCN Resnet101, Faster RCNN Inception v2, YOLO v5, and SSD Mobilenet v2 classifiers. Each model is pre-trained using the coco (Common Objects in Context) dataset 2018 Furthermore, for each classifier, we did hyperparameter tuning and chose the best hyperparameters for each model then we compare the models in terms of recall, precision and number of predicted bounding boxes(detected objects) per image, our goal is around 46 bounding box (number of chromosomes in humans) for each image. Selecting the best hyperparameters is crucial for improving the number of predicted bounding boxes per image. For example, selecting a small epoch number will result in less detected bounding boxes. Moreover, selecting too large batch size might result in longer model training time and less recall and precision. The hyperparameters we changed were learning rate, number of epochs, IOU_threshold and batch size.

4.2 Implementation & Tools

In this section, we discuss different tools and platform that we used to do different tasks. The project has different phases.First, we did data preparation and labelling for chromosome images to classify chromosome classes using *LabelIMG*. Then, we did data augmentation using python and *Roboflow*. Finally, we prepared the machine learning model for training our chromosome dataset using *Tensorflow* on *Google co-laboratory* which provides free GPU for training the model.

4.2.1 LabelIMG

LabelIMG [23] is one of the most widely used tools for image annotation and labelling. It is written in Python and uses Qt for its graphical interface. It runs on Linux, Windows and macOS. The tool has a nice and simple user interface to make it easy and fast to label your data. The tool also enables developers to save annotations as *XML* files, *PASCAL VOC*, and *YOLO* format.

4.2.2 Roboflow

Roboflow is a web-based tool that is used to help engineers working on computer vision and machine learning projects to organise their data and do several different pre-processing operations on the images before training the model. This will save a lot of money and CPU/GPU processing power. It saves over 10% by pre-processing and performing pre-computed augmentation on CPU beforehand. Roboflow also saves a lot of time spent on writing code to do data processing, conversion, augmentation and organization of the datasets. Another advantage of roboflow is that teams can save their datasets in one place, hence, data is kept up-to-date giving the users the ability to export the data in many formats such as JSON, TXT, XML, CSV and TFRecords(For Tensorflow). Finally, Roboflow allows you to export your data directly to your python notebook using a single command.

Roboflow provides multiple pre-processing options, these options include Auto Orientation(Discarding EXIF rotations and standardize pixel ordering), image Resizing, grey-scaling, Contrast adjustment. It also provides some augmentation options such as image flipping, rotation, random shear, changing brightness exposure, blurring and adding random noises.

4.2.3 Tensorflow

We used *Tensorflow* as my machine learning platform. It is an open-source software library for data analytics and deep learning purposes. It offers hundreds of libraries that are specifically made for implementing deep learning models. Tensorflow is written in python and was developed at Google in November 2015.
In this project, we developed a Faster RCNN Inception v2 model, RFCN Resnet101, SSD Mobilenet v2 and YOLO v5 models for chromosome detection using Tensorflow version 1.15.

4.2.4 Google co-laboratory

we used the GPU resources of Google Co-laboratory (Colab) for running the neural networks. Instead of running the model on PC resources, Colab has the same format as Jupyter notebooks and it provides free 70GB disk space, 13GB RAM and GPU resources to run machine learning models for up to 12 hours. The GPU I used was Tesla P100-PCIE-16GB which has a memory of 16GB which is much faster than my local GPU speed. Colab saves to Google Drive which allows users to share and have multiple people work on the same document at once which allows easier code collaboration.

4.3 A look into the dataset

In this section, we discuss the dataset we used and we describe the data augmentation methods that we used to increase the number of images as our dataset was only 75 images which are very low and will typically result in underfitting.

Our dataset is composed of 75 original chromosome images and 75 Karyogram images. Each image consists of 23 pairs of chromosomes or 46 chromosomes in total. Each chromosome typically belongs to a class from 1 to 23 (The 23rd class is usually an X and a Y or 2 X chromosomes). Originally the images were raw so we had to label it and we used the tool LabelIMG to label the data manually as shown in figure 4.2. Then we developed scripts that do data augmentation to generate new images in order to improve the model prediction, reduce class undersampling and reduce underfitting. We generated 832 images with multiple data augmentations such as 180, 90 degree and random rotations, reflections, change in brightness, noise injections, and random cropping as shown in figures 4.3 and 4.4. More information about the class distribution of our dataset can be found in figure 4.5. Finally, our dataset was divided into 583 test, 166 validation and 83 Test images as shown in figure 4.6.

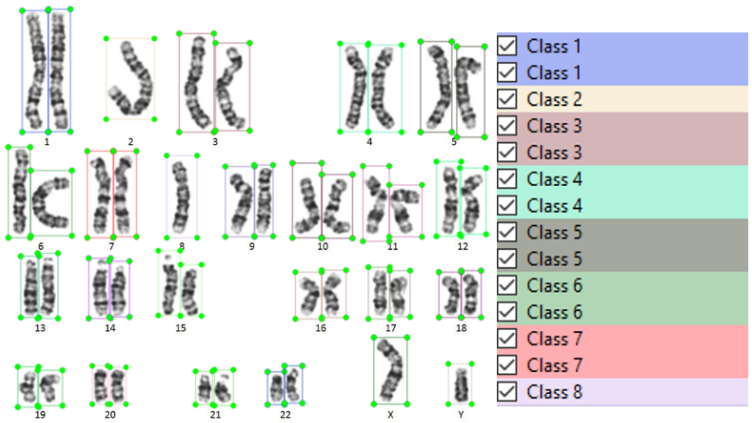

Figure 4.2: Example of labeled data

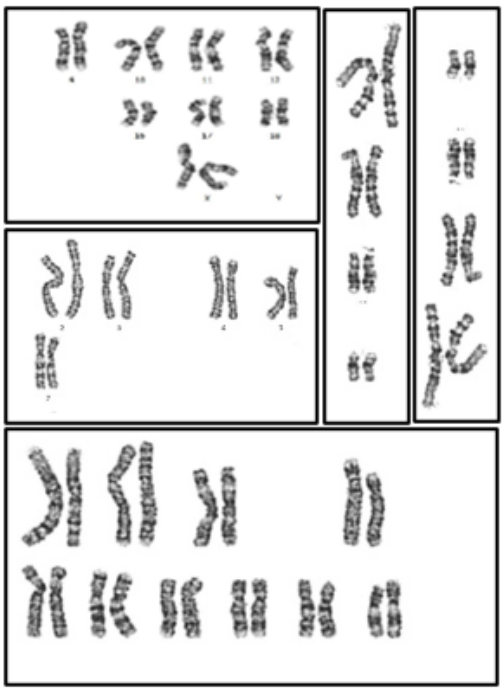

Figure 4.3: Examples of random cropping to emulate real chromosomes for testing our model

Figure 4.4: Examples of images generated from our data augmentation functions using Python

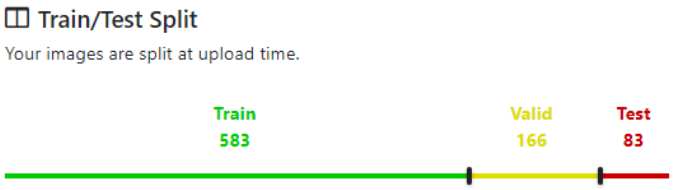

Figure 4.6: Train,test, and validate split

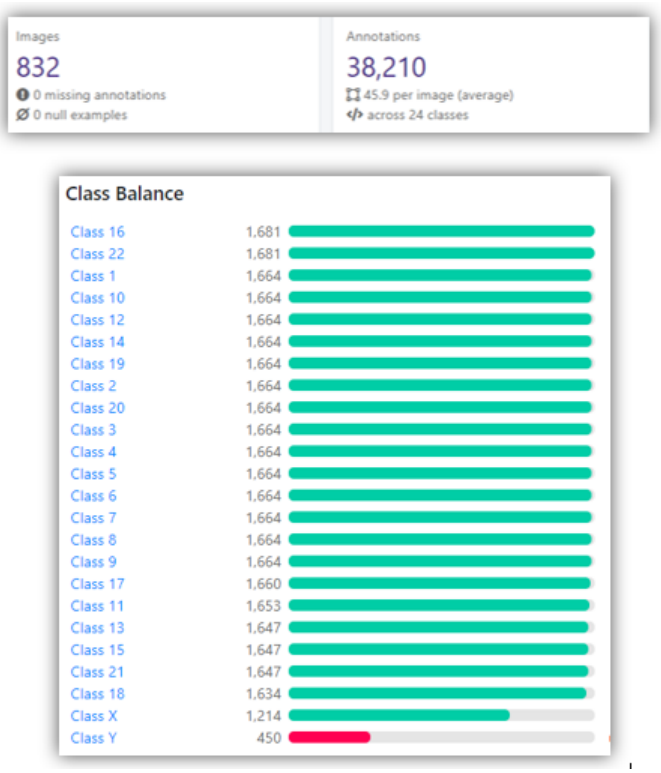

Figure 4.5: Class distribution of our dataset, the diagram shows a similarity in the number of classes from class 1 to 22, an undersample of class X and class Y.

Chapter 5

Performance Evaluation of classification Models

In this chapter we present the results of our experiment and correspondingly, we discuss those results. First, we describe the performance metrics that we used in more details and how we selected the best model for solving our problem. Secondly, we evaluate the results and mention the hyperparameters that achieved the best results.

5.1 Performance Metrics

The metric employed to evaluate a classifier is crucial for determining whether the model is good or not. To evaluate a model, choosing the wrong metric could result in choosing the wrong model or have a wrong holistic view of the expected performance of the model in real-life scenarios. In our problem, we are performing chromosome detection of augmented Karyogram images (which emulate real chromosomes) in order to identify chromosomes and the class of each one of them. We are dealing with a problem which is the under representation of some classes that are lesser than others such as classes X and Y. This will largely affect the choice of evaluation of performance metrics. Let's take for example our case, we have about 1200 classes of X and 420 class Y in our dataset. Thus, the choice of our performance metrics should give us detailed information about our classification models in predicting classes 1-22, X and Y. Consequently, the performance metrics used for the evaluation of our models are- Precision, Recall, and F1 score. F1 score is of particular interest here as described in section 2.2.2, it provides a good overview of the classification algorithms in terms of how good it is in predicting the classes of

each chromosome which are the main interest in our experiment.

We list below the results obtained after running cross-validation using our data and we used several different hyperparameters and we chose the best-performing ones. Subsequently, we recorded the metrics- precision, recall, F1 score of each model. Furthermore, in figures 5.1, 5.2, 5.3 and 5.4 we present the predicted images with bounding boxes of our models and the confidence scores of each prediction. The figures show the predictions using the models Faster RCNN Inception v2, RFCN Resnet101, SSD Mobilenet v2 and YOLO v5 respectively with epoch number of 5000 for all models except YOLO v5 which performed better with less epoch number which was only 200, more than 200 resulted in a very slight improvement and it takes a long time for the model to train.

Figures 5.6, 5.7, and 5.8 presents the recall, precision and F1 scores of the YOLO model which outperformed all other models in terms of recall, precision, F1 as well as the number of bounding boxes per image. Figures 5.9, 5.11 and 5.11 shows the average recall, precision and prediction image for RFCN Resnet. RFCN Resnet. Figures 5.13, 5.12, and 5.2 shows the average recall, precision and prediction image for Faster RCNN Inception v2. Figures 5.16, 5.15 and 5.3 shows the average recall, precision and prediction image for SSD Mobilenet v2.

Table 5.1 presents and compares the recall, precision and F1 results of each model. The table shows the epoch number and batch size used for each of the models. Batch size plays a big role in how fast the model trains and the number of epochs improves the model results when we increase the number of epochs till a certain level, after that the model does not improve and then we can understand that later on when we perform further tests with the same hyperparameters but different images, we should not increase the epoch number above this level as it will not improve the results and the model will take longer time to complete its training.

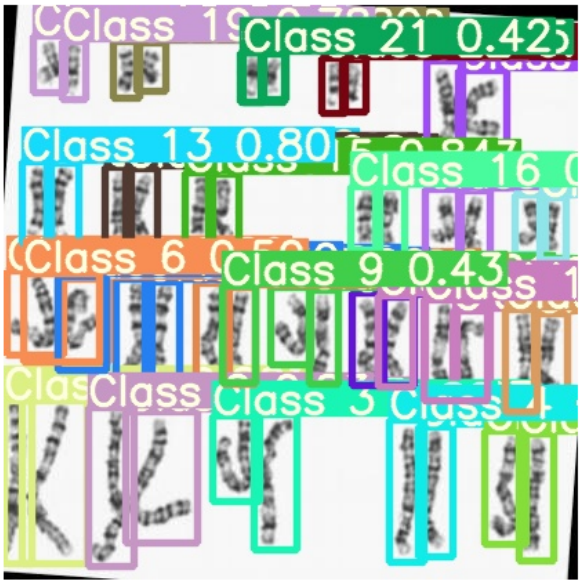

Figure 5.1: Predictions of YOLO v5 model with epoch number of 250.

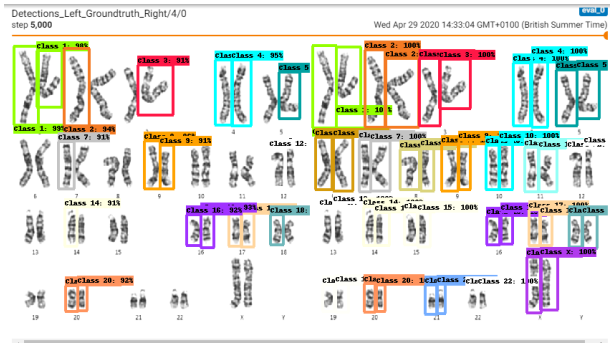

Figure 5.2: Predictions from a test image using Faster RCNN Inception v2.

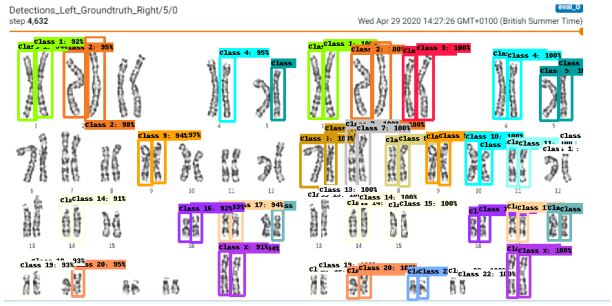

Figure 5.3: Predictions from a test image using SSD Mobilenet v2.

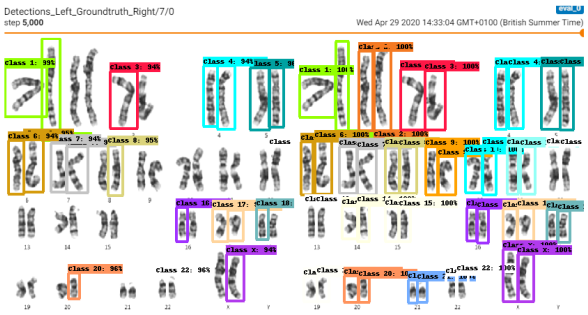

Figure 5.4: Predictions from a test image using RFCN Resnet101.

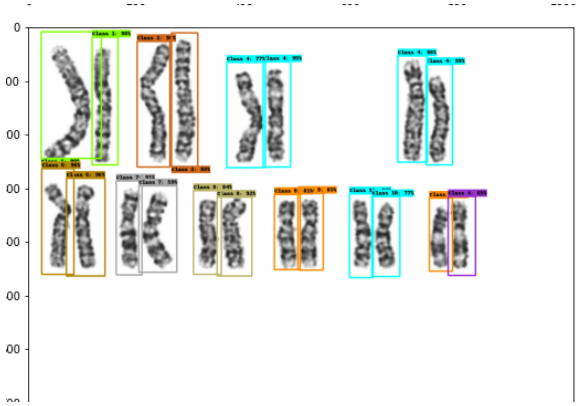

Figure 5.5: Predictions from a test image with only 20 chromosomes using inception v2.

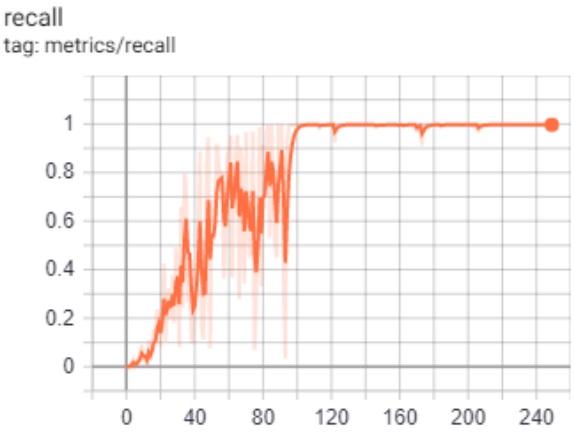

Figure 5.6: Recall YOLO v5. epoch: 250, batch size: 12.

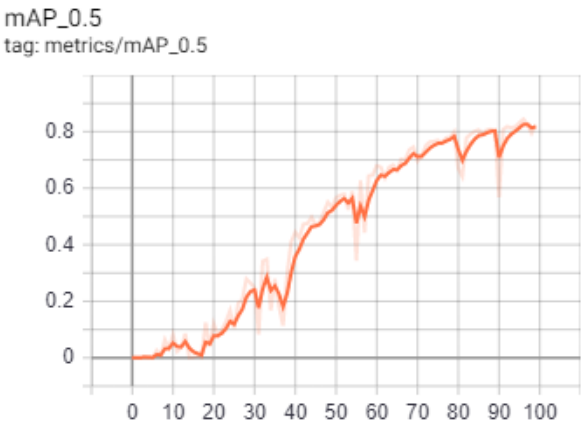

Figure 5.7: mAP YOLO v5 . epoch: 250, batch size: 12.

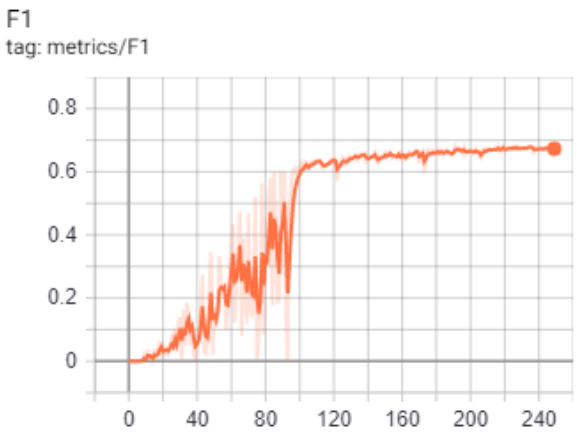

Figure 5.8: F1 Score YOLO v5 . epoch: 250, batch size: 12.

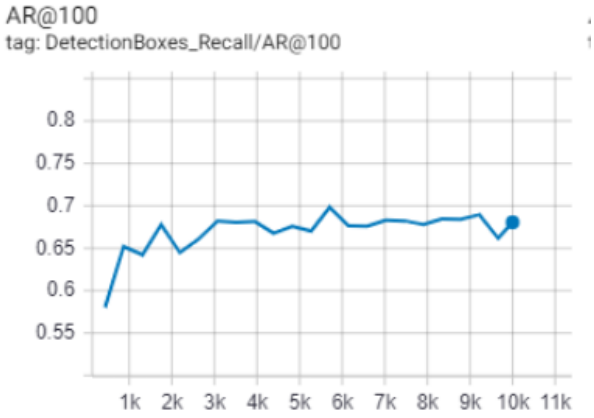

Figure 5.9: Recall RFCN Resnet101. epoch: 10K, batch size: 12, learning rate: 0.0002.

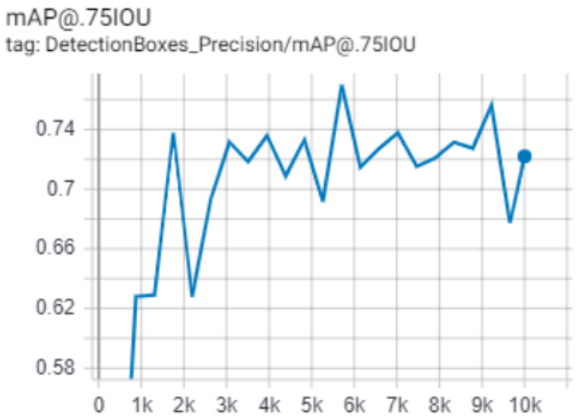

Figure 5.10: mAP RFCN Resnet101. epoch: 10K, batch size: 12, learning rate: 0.0002.

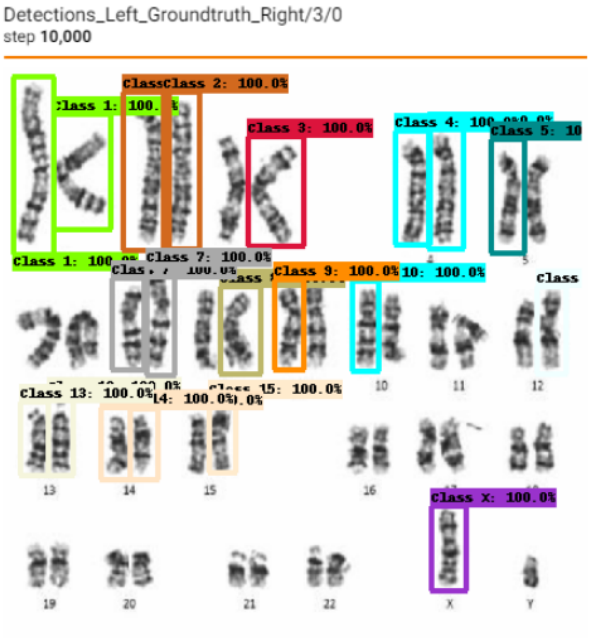

Figure 5.11: Prediction image RFCN Resnet101. epoch: 10K, batch size: 12, learning rate: 0.0002.

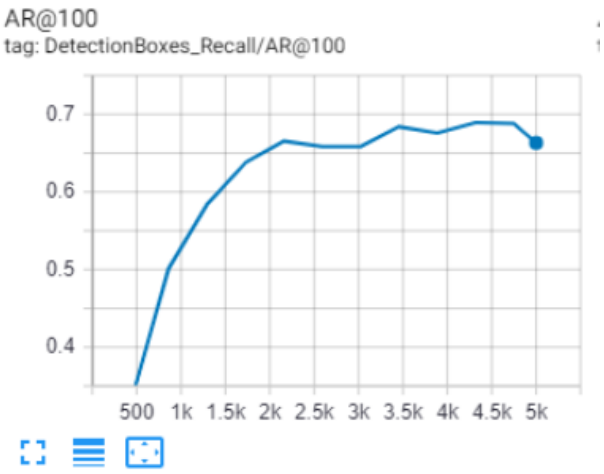

Figure 5.12: Recall of Faster RCNN Inception v2. epoch 5K, batch size: 14, learning rate: 0.0002.

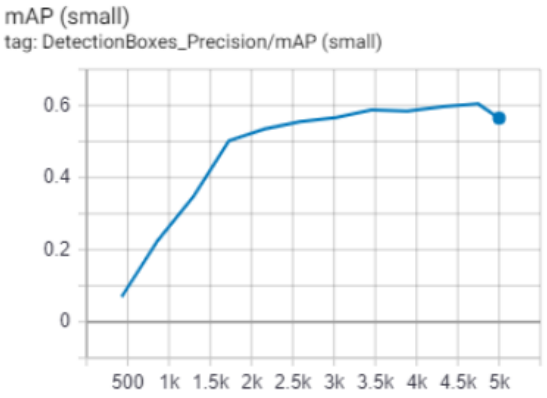

Figure 5.13: mAP Faster RCNN Inception v2. epoch: 5K, batch size: 14, learning rate: 0.0002.

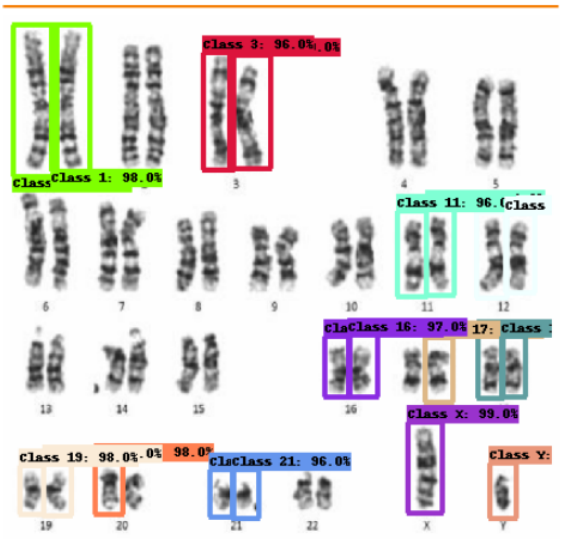

Figure 5.14: Prediction image Faster RCNN Inception v2. epoch: 5K, batch size: 14, learning rate: 0.0002.

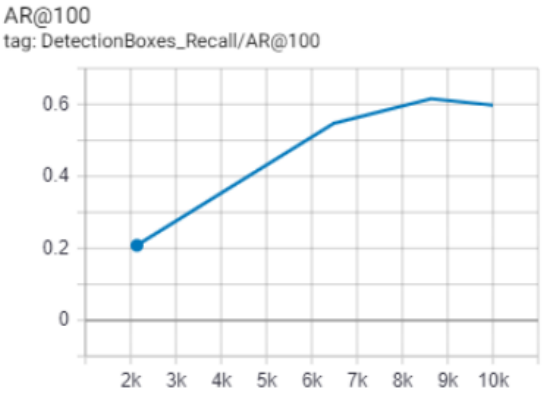

Figure 5.15: Recall SSD Mobilenet v2. epoch: 10K, batch size: 12, learning rate: 0.0002.

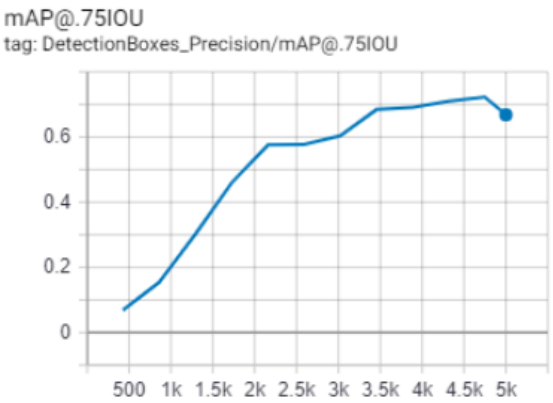

Figure 5.16: mAP SSD Mobilenet v2. epoch: 5K, batch size: 12, learning rate: 0.0002.

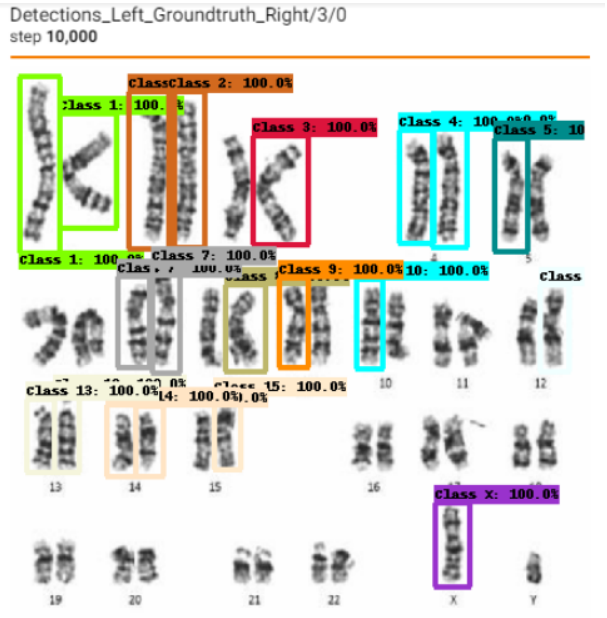

Figure 5.17: Prediction image SSD Mobilenet v2. epoch: 10K, batch size: 12, learning rate: 0.0002.

Model	Batch Size	Epoch No.	Recall	Precision	F1
YOLO v5	16	250	0.98	0.83	0.89
Faster RCNN Inception v2	14	5000	0.66	0.59	0.62
RFCN Resnet101	12	5000	0.68	0.72	0.69
SSD Mobilenet v2	12	10000	0.60	0.69	0.64

Table 5.1: A comparison of the results of our models

5.2 Evaluation of Results

Predictions from models SSD Mobilenet v2, Faster RCNN Inception v2 and RFCN Resnet101 show poor results compared to the ones by YOLO v5 which outperformed all other models in terms of the number of detected chromosomes (bounding boxes) per image. We tested our software for bugs that might be resulting in this worse performance but testing with images with fewer chromosomes resulted in better bounding boxes results as shown in figure 5.5.

Table 5.1 shows that YOLO scored significantly better results with small epoch number, the model was faster, results were more accurate and it detected all chromosomes and predicted the correct classes for each chromosome for

all the test results. The precision score for YOLO was not better than Faster RCNN because we are using a small epoch number, using more images and a higher epoch number will result in a much better result. However, YOLO will take a very long duration to train the model for slight improvement so I decided to keep the epoch number low as long as the model is performing well. On the other hand, the models Faster RCNN, RFCN and SSN showed similar results, after performing many tests, I observed that the model training time is significantly slower than YOLO v5 and it needed more epochs to show good precision and recall results. Furthermore, the number of bounding boxes for these models were limited to 21-24 bounding boxes even after performing many tests with 5000, 6000, 7000, 8000, 9000, and 10000 epochs which took more than 8 hours to run the 10000 epochs for these 3 models without showing any improvement in the recall and precision.

Chapter 6

Conclusion and Future work

We implemented 4 methods, Faster RCNN, RFCN, SSN, and YOLO v5 for chromosome detection and classification to automate the process of karyotyping. First, as we only had 75 images, we performed several image augmentation techniques on our dataset that resulted in generating 832 images to improve the model prediction. Secondly, We trained each model using tens of different hyperparameters to get the best results possible. Finally, We compared the results of each model and we can clearly notice that YOLO v5 the best. Unlike other models, YOLO v5 showed impressive recall, precision and F1 scores, with the scores 0.98, 0.83 and 0.89, respectively. Furthermore, it was able to detect and classify all the 46 chromosomes in the Karyogram dataset. Our approach is a large step towards automating chromosome diagnostics using Artificial Intelligence and machine learning. Other chromosome diagnostics approaches that use machine learning with unorganized chromosome datasets instead of Karyogram datasets have significantly low accuracy scores for detecting and predicting chromosome abnormalities. Thus, we have to first take this step towards automating karyotyping then we can use machine learning for chromosome abnormality detection using the results used for classification.

Bibliography

[1] Gert de Graaf, Frank Buckley, and Brian G Skotko. "Estimates of the live births, natural losses, and elective terminations with Down syndrome in the United States". In: *American Journal of Medical Genetics Part A* 167.4 (2015), pp. 756–767.

[2] Keiron O'Shea and Ryan Nash. "An introduction to convolutional neural networks". In: *arXiv preprint arXiv:1511.08458* (2015).

[3] Francois Chollet. "Deep Learning with Python. 978-1617294433". In: *Shelter Island: Manning Publications* (2017), p. 384.

[4] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. "Artificial neural networks: A tutorial". In: *Computer* 29.3 (1996), pp. 31–44.

[5] M Sornam, Kavitha Muthusubash, and V Vanitha. "A survey on image classification and activity recognition using deep convolutional neural network architecture". In: *2017 Ninth International Conference on Advanced Computing (ICoAC)*. IEEE. 2017, pp. 121–126.

[6] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[7] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[8] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[9] Jasper RR Uijlings et al. "Selective search for object recognition". In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[10] RB Girshick. *Fast R-CNN. CoRR, abs/1504.08083*. 2015.

[11] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[12] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[13] David Martin Powers. “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation”. In: (2011).

[14] Mohammad Tariqul Islam et al. “Abnormality detection and localization in chest x-rays using deep convolutional neural networks”. In: *arXiv preprint arXiv:1705.09850* (2017).

[15] Vikash Chouhan et al. “A Novel Transfer Learning Based Approach for Pneumonia Detection in Chest X-ray Images”. In: *Applied Sciences* 10.2 (2020), p. 559.

[16] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. “A survey of transfer learning”. In: *Journal of Big data* 3.1 (2016), p. 9.

[17] Sivaramakrishnan Rajaraman et al. “A novel stacked generalization of models for improved TB detection in chest radiographs”. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2018, pp. 718–721.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[19] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[20] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[21] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[22] Xingwei Wang et al. “Automated classification of metaphase chromosomes: optimization of an adaptive computerized scheme”. In: *Journal of biomedical informatics* 42.1 (2009), pp. 22–31.

[23] “LabelIMG: a tool for data labeling”. In: 3.1 (2016), p. 9.