**CLASSIFICATION OF CHROMOSOME USING DEEP LEARNING**

**ABSTRACT**

For every 150 babies born, there is one new born baby with a Chromosomal abnormality. The manual segmentation and analysis for each chromosome for diagnosis purposes consumes a considerable amount of time and is highly dependent on expert knowledge.

The primary purpose of our project was to be able to classify all the 23 Chromosomes with a high accuracy using Deep Learning. This would help automate or semi-automate the karyotyping process in order to assist the doctors and reduce their cognitive load. Apart from classifying, we have also been able to detect each strand of the 46 chromosomes.

It  motivates us to automate or semi-automate the karyotyping process in order to assist doctors and reduce their cognitive load. We propose to have an end to end Deep Learning architecture for expediting the task of karyotyping.

Though there are existing models available for increasing the speed of the karyotyping process, their accuracy could only be recorded till 94%. This is where our model was able to perform better and recorded an accuracy of 96.6%.

**ACKNOWLEDGEMENT**

We express our sincere gratitude to **Dr. S. Ganesh Vaidyanathan, Ph.D.**, Principal, Sri Venkateswara College of Engineering, for being the source of inspiration throughout our study in college.

We thank **Dr. S. Muthukumar, Ph.D.**, Professor, Head of the Department, Electronics and Communication Engineering, for his permission and encouragement to carry out this project.

We are extremely grateful to **Ms. D. Menaka, M.E**, Assistant professor, Department of Electronics and Communication Engineering, for her guidance, constant encouragement and support for the successful completion of this project.

We wish to thank our project coordinators **Mrs. K. S. Subhashini, M.E.**, Assistant Professor, **Mr**.**S P Sivagnana Subramanian, M.E,** Assistant Professor, **Dr. S. Sutha M.Tech, Ph.D** **,** Assistant Professor for their valuable inputs and for encouraging us throughout this project.

We also extend our sincere thanks to all teaching and non-teaching staff of the Department of the Electronics and Communication Engineering for their valuable support and guidance towards the completion of the project.

PIYUSH M

RAHIL H SAKARIYA

RAHIL JAIN

**TABLE OF CONTENTS**

**CHAPTER NO.** **TITLE** **PAGE NO**

**ABSTRACT**  3

**ACKNOWLEDGEMENT**  4

**LIST OF FIGURES** 8

**LIST OF TABLES**  10

1. **OVERVIEW**  11

PROJECT FLOW 11

2. **LITERATURE REVIEW** 12

3. **LOCALIZATION**  15

3.1 COCO - ANNOTATOR 15

3.1.1 OVERVIEW 15

3.1.2 FEATURES 16

3.1.3 RESULTS 17

3.2 DETECTRON2 18

3.2.1 OVERVIEW 18

3.2.2 FEATURES 18

3.2.3 RESULT 20

3.3 YOLO V4 21

3.3.1 OVERVIEW 21

3.3.2 ARCHITECTURE 21

3.3.3 RESULT 23

4. **DATA PREPARATION** 24

4.1 DATA SEGREGATION 24

4.2 SUPER RESOLUTION 24

4.2.1 OVERVIEW 24

4.2.2 ENHANCED DEEP RESIDUAL NETWORK 25

4.2.3 EFFICIENT SUBPIXEL

CONVOLUTIONAL NETWORK 26

4.2.4 FAST SUPER RESOLUTION

CONVOLUTIONAL NETWORK 26

4.2.5 LAPLACIAN PYRAMIDAL SUPER

RESOLUTION NETWORK 27

4.2.5.1 FEATURE EXTRACTION 28

4.2.5.2 IMAGE RECONSTRUCTION 29

5. **PROPOSED METHOD** 31

5.1 COMPUTE AND TRAINING TOOLS 31

5.1.1 GPU 31

5.1.2 COLAB 32

5.2 METRICS 32

5.2.1 PRECISION 33

5.2.2 RECALL 33

5.2.3 F1 SCORE 33

5.3 PROPOSED MODEL 34

5.3.1 CUSTOM MODEL 34

5.3.2 RESNETS 36

5.3.3 VGG NET 36

5.3.4 EFFICIENT NET 37

5.3.4.1. OVERVIEW 37

5.3.4.2. SWISH ACTIVATION 39

5.3.4.3. BASIC STRUCTURE 39

5.3.4.4. B0 MODEL 41

5.3.4.5. B5 MODEL 42

5.3.4.6. B6 MODEL 43

5.3.5 CLASSIFICATION REPORT 44

5.3.6 MODEL SUMMARY 45

5.3.7 CALCULATION 46

5.3.8 COMPARISON 46

6.  **FUTURE WORK**  47

6.1. ABNORMALITY DETECTION 47

6.1.1 NUMERICAL ABNORMALITIES 47

6.1.2 STRUCTURAL ABNORMALITIES 47

7. **CONCLUSION**  50

**REFERENCES**  51

**LIST OF FIGURES**

FIGURE NO. TITLE PAGE NO.

1.1 PROJECT FLOW 11

3.1.1 MANUALLY SEGMENTATION OF CHROMOSOME 16

3.1.2 ANNOTATED IMAGES OF CHROMOSOME 17

3.2.1 EXAMPLE OF DETECTRON MODEL 19

3.2.2 CHROMOSOME IDENTIFIED 20

3.2.3 AP VALUES 20

3.3.1 ARCHITECTURE OF YOLOV4 MODEL 21

3.3.2 OUTPUT OF YOLOV4 MODEL 23

4.2.1 ARCHITECTURE OF ENHANCED DEEP RESIDUAL 25

NETWORKS

4.2.2 ARCHITECTURE OF EFFICIENT SUBPIXEL 26

CONVOLUTIONAL NETWORK

4.2.3 ARCHITECTURE OF FSRCNN 27

4.2.4 ARCHITECTURE OF LAPLACIAN PYRAMIDAL SUPER 28

RESOLUTION NETWORK

4.2.5 OUTPUT FOR LAPLACIAN PYRAMIDAL SUPER 29

RESOLUTION

4.2.6 IMAGE BEFORE SUPER RESOLUTION 30

4.2.7 IMAGE AFTER SUPER RESOLUTION 30

5.3.1 CNN LAYER - CUSTOM MODEL 34

5.1.2 SEQUENTIAL MODEL OF CNN ARCHITECTURE 35

5.2.1 ARCHITECTURE OF RESNET 36

5.3.1 ARCHITECTURE OF VGG NET 36

5.4.1 DIFFERENT SCALING METHOD FOR EFFICIENTNET 37

5.4.2 EFFICIENTNET VS RESNET 38

5.4.3 BASE STRUCTURE FOR EFFICIENTNET 40

5.4.4 ARCHITECTURE OF EFFICIENTNET B0 MODEL 41

5.4.5 ARCHITECTURE OF EFFICEINTNET B5 MODEL 42

5.4.6 ARCHITECTURE OF EFFICIENTNET B6 MODEL 43

**LIST OF TABLES**

TABLE NO. NAME PAGE NO

5.4.1 CLASSIFICATION REPORT ON B6 MODEL 44

5.4.2 MODEL SUMMARY 45

5.4.3 COMPARISON WITH STATE OF THE ART

MODELS 46

**CHAPTER 1**

**OVERVIEW**

This report discusses the result of the work done in development of an end to end Deep Learning Architecture for detecting and classifying chromosomes. The aim of this project was to obtain a significant higher accuracy for classifying and detecting chromosomes than the current existing architectures present at the time.

**PROJECT FLOW**

The project structure can be split into 3 parts:

* Localization
* Classification
* Abnormality Detection



Fig: 1.1 Project Flow

**CHAPTER 2**

**LITERATURE REVIEW**

The impact of Deep Learning in the medical society is truly remarkable. There are many state of the art deep learning models which have performed exceptionally well in Detection of Chromosomes in aid to speed the Karyotyping process and to detect any abnormalities.

In Jan 2019, Swati, Monika Sharma, and Lovekesh Vig proposed a paper “Automatic Classiﬁcation of Low-Resolution Chromosomal Images”. In this paper, they propose a Super-Xception network which takes the low-resolution chromosome images as input and classiﬁes them to one of the 24 chromosome class labels after conversion into high res-olution images. In this network, they integrate super-resolution deep models with standard classiﬁcation networks e.g., Xception network. The network is trained in an end-to-end manner in which the super-resolution layers help in conversion of low-resolution images to high-resolution images which are subsequently passed through deep classiﬁcation layers for label assigning. They evaluate the proposed network’s efﬁcacy on a publicly available online Bioimage chromosome classiﬁcation dataset of healthy chromosomes and benchmark it against the baseline models created using traditional deep convolutional neural network, ResNet-50 and Xception network.

In July 2018, Swati, Monika Sharma, and Lovekesh Vig proposed a paper “Automatic Chromosome Classification using Deep Attention Based Sequence Learning of Chromosome Bands”. They proposed Residual Convolutional Recurrent Attention Neural Network (Res-CRANN) which exploits this property of band sequence for chromo-some classification. Res-CRANN is end-to-end trainable in which a sequence of feature vectors, extracted from the feature maps produced by convolutional layers of Residual neural networks (ResNet) is fed into Recurrent Neural Networks (RNN) and subsequently, an attention mechanism is applied on top of RNN output sequences which are further classified into one of the 24 labels. The attention mechanism after recurrent layers facilitates the network to learn to pay selective attention to the sequence of bands and relate them to different classes of chromosomes. They demonstrate the proposed architecture's efficacy on a publicly available Bioimage chromosome classification dataset and observe that the model outperforms the baseline models created using traditional deep convolutional neural network and ResNet-50 by approximately 3% Top-1 classification accuracy.

In July 2020, MONA SALEM AL-KHARRAZ, LAMIAA A. ELREFAEI AND MAI AHMED FADEL proposed a paper “Automated System for Chromosome Karyotyping to Recognize the Most Common Numerical Abnormalities Using Deep Learning ”. In this research, they used deep learning to automate karyotyping to recognize the common numerical abnormalities on a dataset containing 147 non-overlapped metaphase images collected from the Center of Excellence in Genomic Medicine Research at King Abdulaziz University. The metaphase images went through three stages. The first one is individual chromosomes detection using YOLOv2 Convolutional Neural Network followed by some chromosome post-processing. This step achieved 0.84 mean IoU, 0.9923 AP, and 100% individual chromosomes detection accuracy. The second stage is feature extraction and classification where they fine-tune the VGG19 network using two different approaches, one by adding extra fully connected layer(s) and another by replacing fully connected layers with the global average pooling layer. The best accuracy obtained is 95.04%. The final step is detecting abnormality and this step obtained 96.67% abnormality detection accuracy. To further validate the proposed classification method, they examined the Biomedical Imaging Laboratory dataset which is publicly available online and achieved 94.11% accuracy

In Feb 2020, Yirui Wu, Xiao Tan and Tong Lu proposed a paper “A New Multiple-Distribution GAN Model to Solve Complexity in End-to-End Chromosome Karyotyping”. They firstly utilize visual features to generate chromosome candidates with Extremal Regions (ER) technology. Due to severe occlusion and cross overlapping, they utilize ring radius transform to cluster pixels with same property to approximate chromosome shapes. To solve the problem of unbalanced and small dataset by covering diverse data patterns, they proposed multidistributed generated advertising network (MD-GAN) to perform data enhancement by generating additional training samples. Afterwards, they fine-tune CNN for chromosome classification task by involving generated and sufficient training images. Through experiments in self-collected datasets, the proposed method achieves high accuracy in tasks of chromosome detection, segmentation, and classification. Moreover, experimental results prove that MD-GAN-based data enhancement contributes to classification results of CNN to a certain extent.

**CHAPTER 3**

**LOCALIZATION**

**3.1 COCO ANNOTATOR**

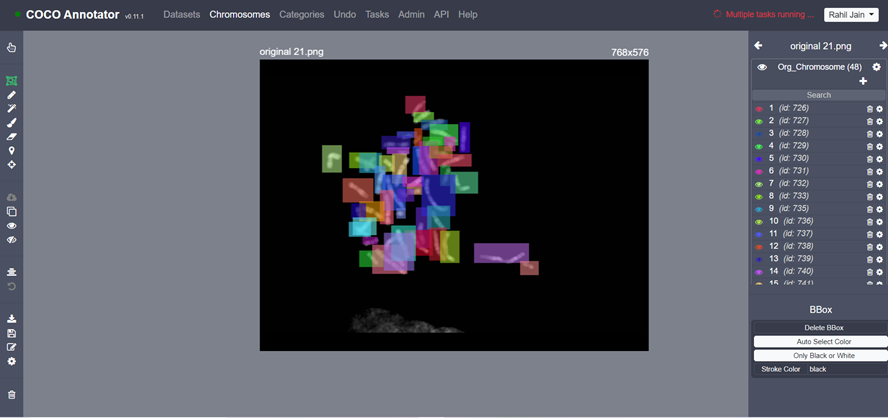
**3.1.1. OVERVIEW**

COCO ANNOTATOR is a web based annotation tool used for image segmentation. The annotation process is delivered through an intuitive and customizable interface and provides many tools for creating accurate datasets. It provides many distinct features including the ability to label an image segment (or part of a segment), track object instances, labeling objects with disconnected visible parts, efficiently storing and export annotations in the well-known COCO format.

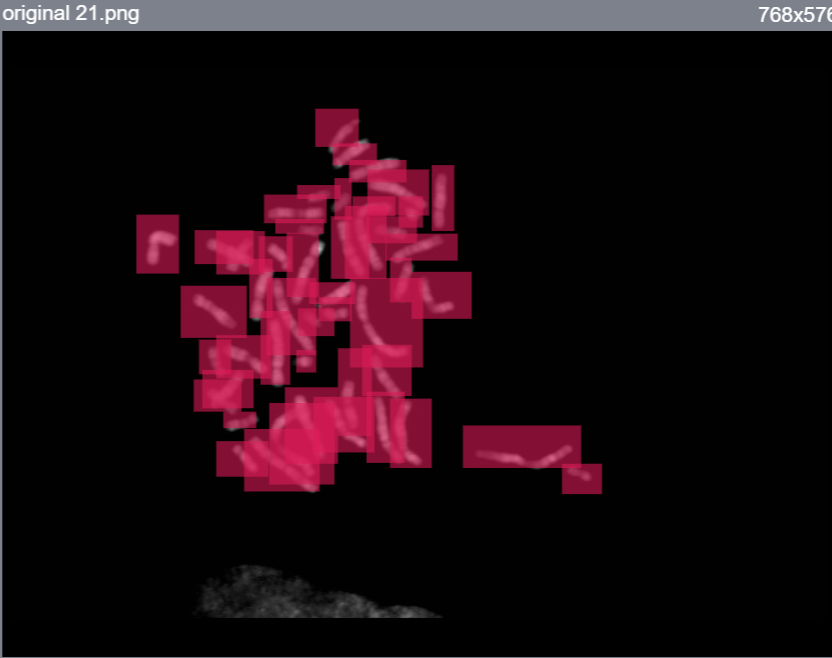
**3.1.2. FEATURES**

*COCO Annotator* allows users to annotate images using free-form curves or polygons and provides many additional features where other annotation tools fall short.

* Directly export to COCO format
* Segmentation of objects
* Ability to add key points
* Useful API endpoints to analyze data
* Import datasets already annotated in COCO format
* Annotate disconnect objects as a single instance
* Labeling image segments with any number of labels simultaneously
* Allow custom metadata for each instance or object
* Advanced selection tools such as,DEXTR, MASK-RCNN and Magic Wand
* Annotate images with semi-trained models
* Generate datasets using google images
* User authentication system

(Fig 3.1.1 Manually segmenting all the chromosome)

We have used Docker applications to run COCO-Annotator. The User Interface of the application can be seen in the above figure.



(Fig 3.1.2 Annotated Images of Chromosomes)

**3.1.3. RESULTS**

We have also annotated 62 images of Chromosome. Through this segmentation we were able to get 2480 individual chromosome strands. These individual strands will be used as training and testing data for the Detectron2 model.

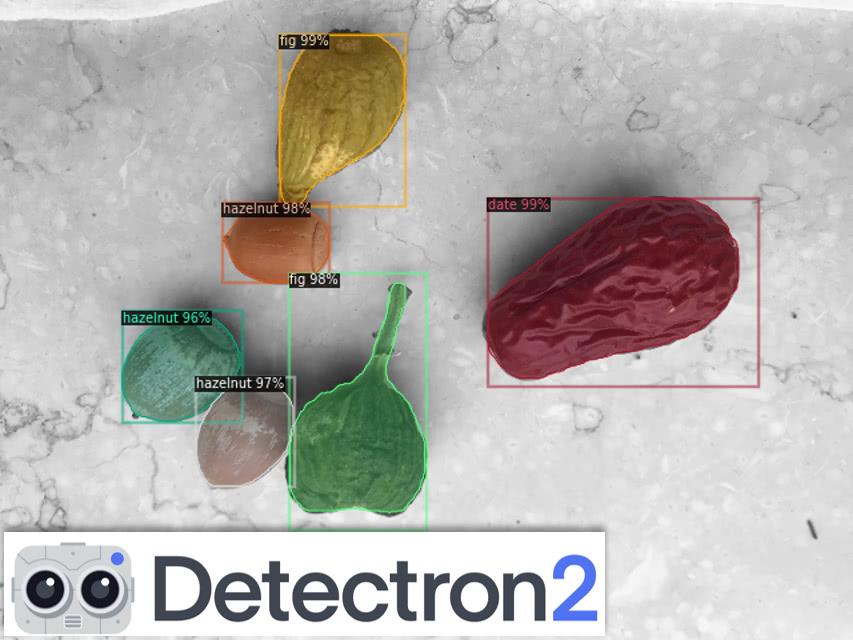
**3.2. DETECTRON 2**

**3.2.1 OVERVIEW**

Detectron2 was developed by FAIR (Facebook Artificial Intelligence Research). It is a framework where we can use state-of-the-art models suchs as Fast, Faster and Mask-R-CNN with ease. It is used to train custom datasets and detect objects. All the models present in the model zoo of the Detectron2 library are pre-trained on COCO Dataset.

**3.2.2 FEATURES**

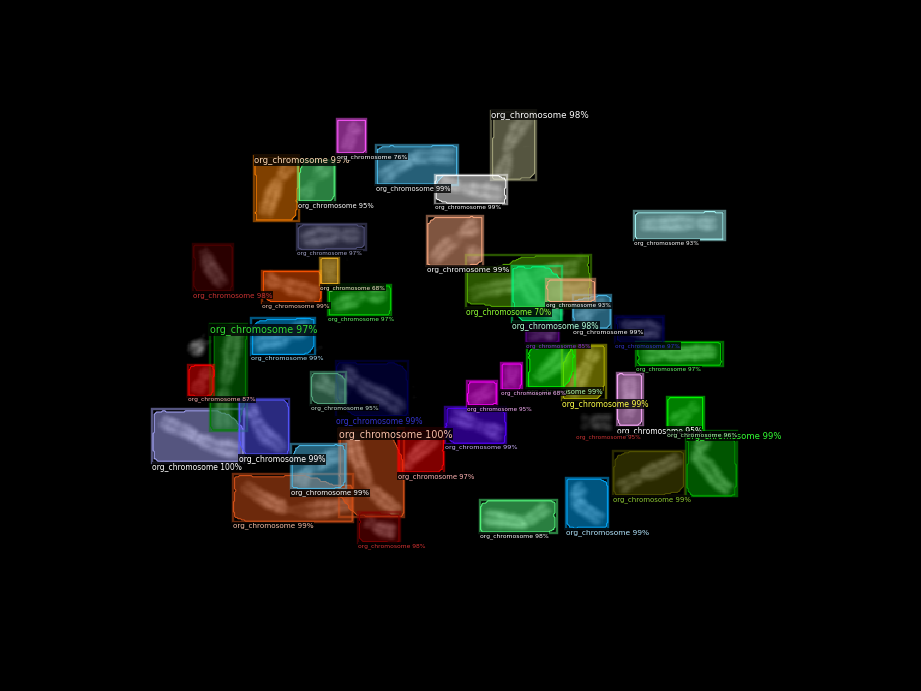
* This time it is Powered by Pytorch deep learning framework.
* Panoptic segmentation
* Include Densepose
* Provide a wide set of baseline results and trained models for download in the Detectron2 model zoo.
* Included projects like DeepLab, TensorMask, PointRend, and more.
* Can be used as a wrapper on top of other projects.
* Exported to easily accessible formats like caffe2 and **torchscript**.
* Flexible and fast training on single or multiple GPU servers



(Fig 3.2.1 Example for Detectron Model)

As you can see in Fig 1.2.1, the Detectron model will detect each fruit. It will add a bounding box on it and show the name of the fruit with its accuracy.

**3.2.3. RESULT**



(Fig 3.2.2 Chromosome Identified)

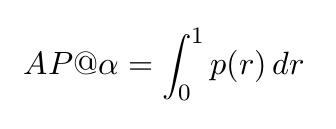
| AP | AP50 | AP75 | APs | APm | APl |

|:------:|:------:|:------:|:------:|:------:|:------:|

| 66.758 | 84.719 | 79.685 | 23.917 | 54.933 | 79.514 |

(Fig 3.2.3 AP values)

AP@α is, ideally, the Area Under the PR Curve (AUC-PR). Mathematically, AP is defined as

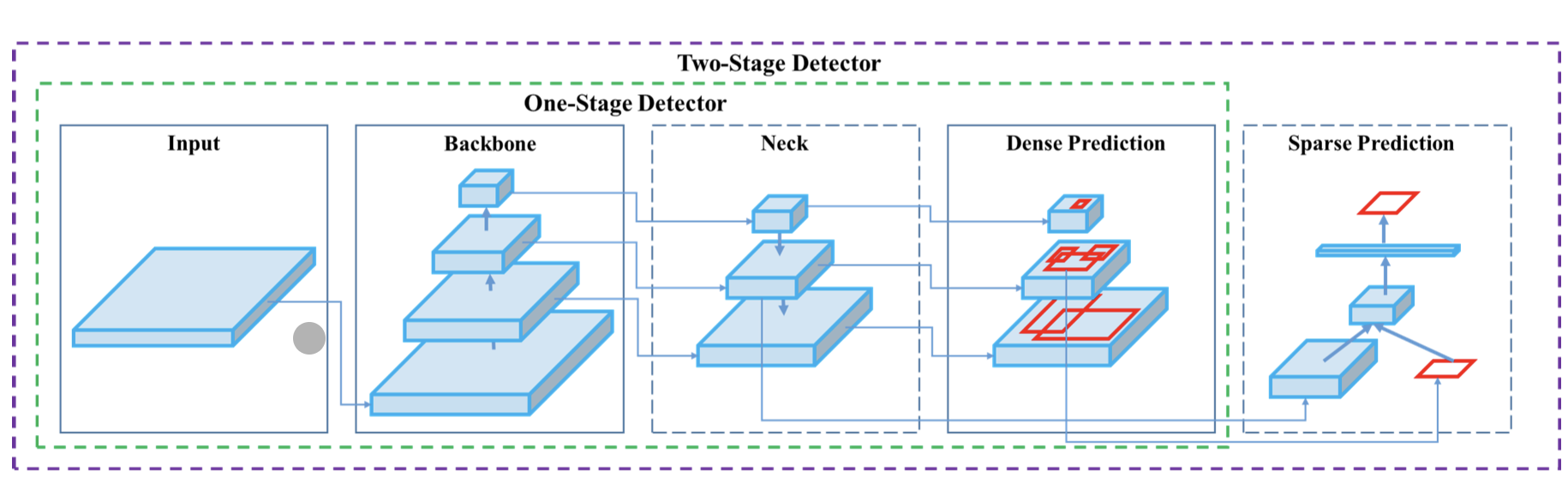


Our model will detect and segment each Chromosome. These segmented images will then go to out classification model (EfficientNet) to classify all the 24 Chromosomes.

**3.3. YOLO V4**

**3.3.1. OVERVIEW**

You only look once (YOLO) is a family of one-stage object detectors that are fast and accurate. Yolo V4 is an important improvement of YoloV3, the implementation of a new architecture in the **Backbone** and the modifications in the **Neck** have improved the **mAP**(mean Average Precision) by **10%** and the number of **FPS**(Frame per Second) by **12%**. In addition, it has become easier to train this neural network on a single GPU.

**3.3.2ARCHITECTURE**

(Fig 3.3.1 Architecture of YOLO V4 Model)

**BACKBONE**

Models such as ResNet,DenseNet, VGG, etc, are used as feature extractors. They are pre-trained on image classification datasets, like ImageNet, and then fine-tuned on the detection dataset. Turns out that these networks that produce different levels of features with higher semantics as the network gets deeper (more layers), are useful for latter parts of the object detection network.

**NECK**

These are extra layers that go in between the backbone and head. They are used to extract different feature maps of different stages of the backbone.

**DENSE PREDICTION**

The dense prediction is the final prediction which is composed of a vector containing the coordinates of the predicted bounding box (center, height, width), the confidence score of the prediction and the label.

**3.3.3. RESULT FOR YOLOV4 MODEL**



(Fig 3.3.2 Output of Object Detection by YOLO V4)

Compared to our Detectron 2 model, YOLO V4 gives a comparatively less accuracy. It is also not able to detect each Chromosome from the image.

**CHAPTER 4**

**DATA PREPARATION**

**4.1. DATA SEGREGATION**

To classify the Chromosomes, we had gathered over 5000 images on all 24 different Chromosomes. Then each image out of the 5000 images was segmented based on their label (Chromosome number).

The images which we gathered were in .bmp format and it was changed to .png format. This was done so that our classification model can detect images more easily in .png format than .bmp format.

**4.2. SUPER RESOLUTION**

**4.2.1 OVERVIEW**

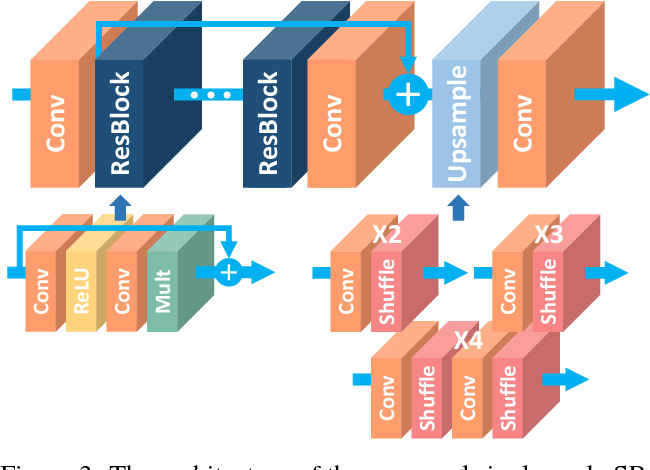
OpenCV is an open-source computer vision library that has an extensive collection of great algorithms. Since one of the latest mergers, OpenCV contains an easy-to-use interface for implementing Super Resolution (SR) based on deep learning methods. The interface contains pre-trained models that can be used for inference very easily and efficiently

There are 4 models in Super Resolution:

* + Enhanced Deep Residual Network
  + Efficient Subpixel Convolutional Network
  + Fast Super-Resolution Convolutional Neural Networks
  + Laplacian Pyramid Super-Resolution Network

**4.2.2. ENHANCED DEEP RESIDUAL NETWORK:**

A enhanced **deep residual network** is a type of specialized neural **network** that helps to handle more sophisticated **deep** learning tasks and models. It has received quite a bit of attention at recent IT conventions, and is being considered for helping with the training of **deep networks**.

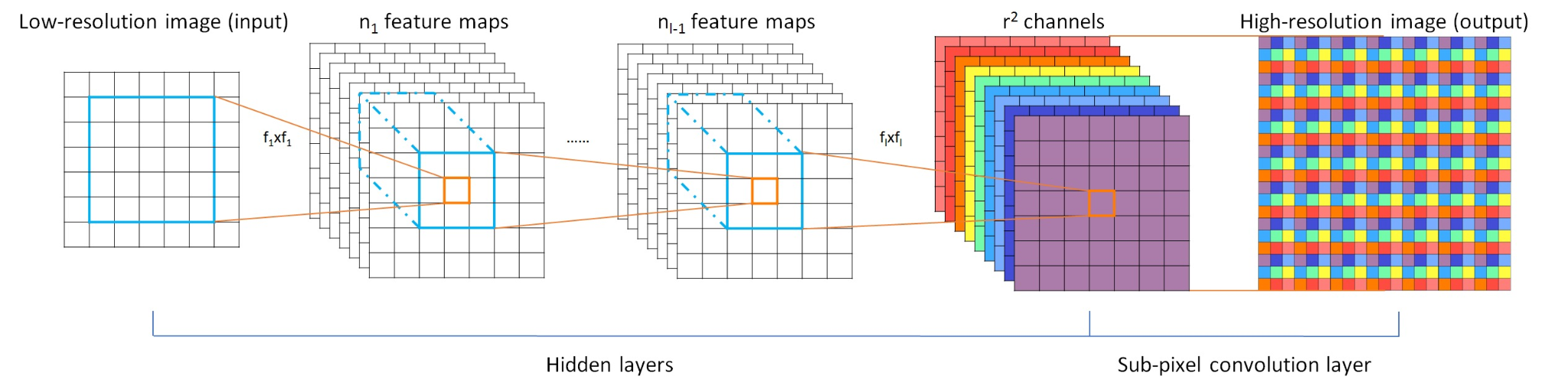


(Fig 4.2.1 Architecture of Enhanced Deep Residual Network)

General CNN architecture with depth (the number of layers) *B* and width (the number of feature channels) *F* occupies roughly *O*(*BF*) memory with *O*(*BF²*) parameters.Therefore, increasing *F* instead of *B* can maximize the model capacity when considering limited computational resources.However, increasing the number of feature maps make the training unstable.Residual scaling with factor 0.1 is applied at the residual path before adding back to the convolutional path. In each residual block, constant scaling layers are placed after the last convolution layers. In the test phase, this layer can be integrated into the previous convolution layer for the computational efficiency.There is no ReLU activation layer outside the residual blocks.

**4.2.3. EFFICIENT SUBPIXEL CONVOLUTIONAL NETWORK:**

ESPCN (Efficient Sub-Pixel CNN) is a model that reconstructs a high-resolution version of an image given a low-resolution version. It leverages efficient "sub-pixel convolution" layers, which learns an array of image upscaling filters.

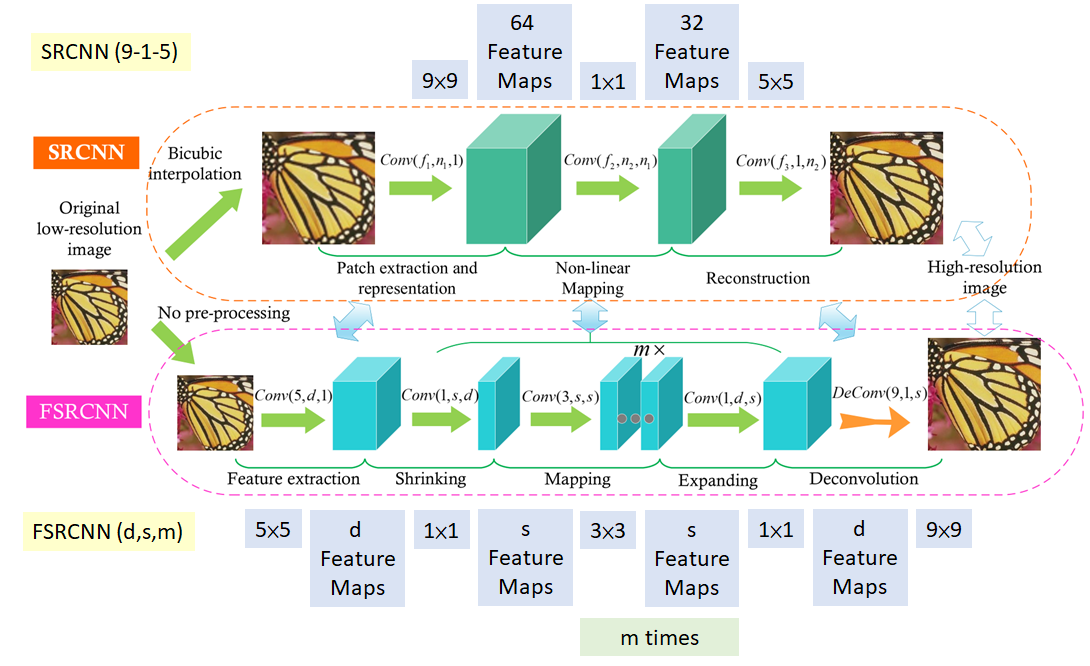


(Fig 2.2.2 Architecture of Efficient Subpixel Convolutional Network)

Upscaling at the last layer. Network operations like feature extracting are done at low resolution space. This means that this network needs less computational resources than operating network at high resolution.

**4.2.4. FAST SUPER RESOLUTION CONVOLUTIONAL NETWORK**

FSRCNN has a relatively shallow network which makes it easier to learn about the effect of each component. It is even faster with better reconstructed image quality than the previous SRCNN.



(Fig 4.2.3 Architecture of FSRCNN)

In FSRCNN, 5 main steps as in the figure with more convolutions are involved:

**Feature Extraction**: Bicubic interpolation in previous SRCNN is replaced by 5×5 conv.

**Shrinking**: 1×1 conv is done to reduce the number of feature maps from d to s where s<<d.

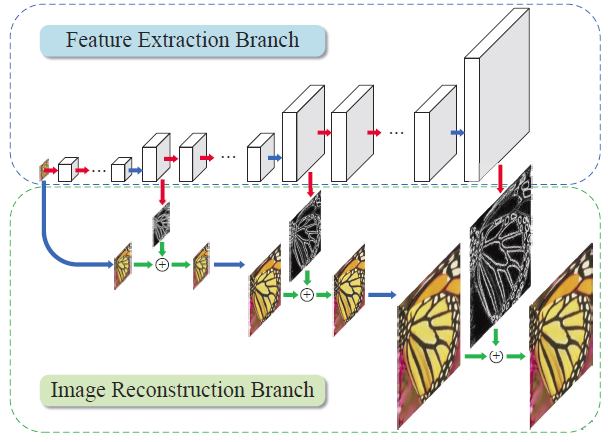
**Non-Linear Mapping**: Multiple 3×3 layers are to replace a single wide one

**Expanding**: 1×1 conv is done to increase the number of feature maps from s to d.

**Deconvolution**: 9×9 filters are used to reconstruct the HR image.

**4.2.5. LAPLACIAN PYRAMID SUPER RESOLUTION STRUCTURE**

By progressively reconstructing the sub-band residuals, with Charbonnier loss functions, LapSRN outperforms SRCNN, FSRCNN, VDSR, and DRCN. With parameter sharing, local residual learning (LRL) and multi-scale training, MS-LapSRN even outperforms DRRN.



(Fig 4.2.4 Architecture of Laplacian Pyramid Super Resolution Structure)

In contrast to one-step upsampling, the network progressively reconstructs the sub-band residuals of high-resolution images at multiple pyramid levels, specifically at log2(*S*) levels where *S* is the scale factor (i.e., 2, 4, 8).

Without the use of bicubic, it directly extracts features from the low-resolution input image, and thereby low computational loads.

There are two branches: Feature extraction & Image Reconstruction.

## **4.2.5.1. FEATURE EXTRACTION**

At level *s*, there are *d* convolutional layers and one transposed convolutional layer (Or deconvolutional layer) to upsample the extracted features by a scale of 2.

The output of each transposed convolutional layer is connected to two different layers: (1) a convolutional layer for reconstructing a residual image at level *s*, and (2) a convolutional layer for extracting features at the finer level *s*+1.

The feature representations at lower levels are shared with higher levels, and thus can increase the non-linearity of the network to learn complex mappings at the finer levels.

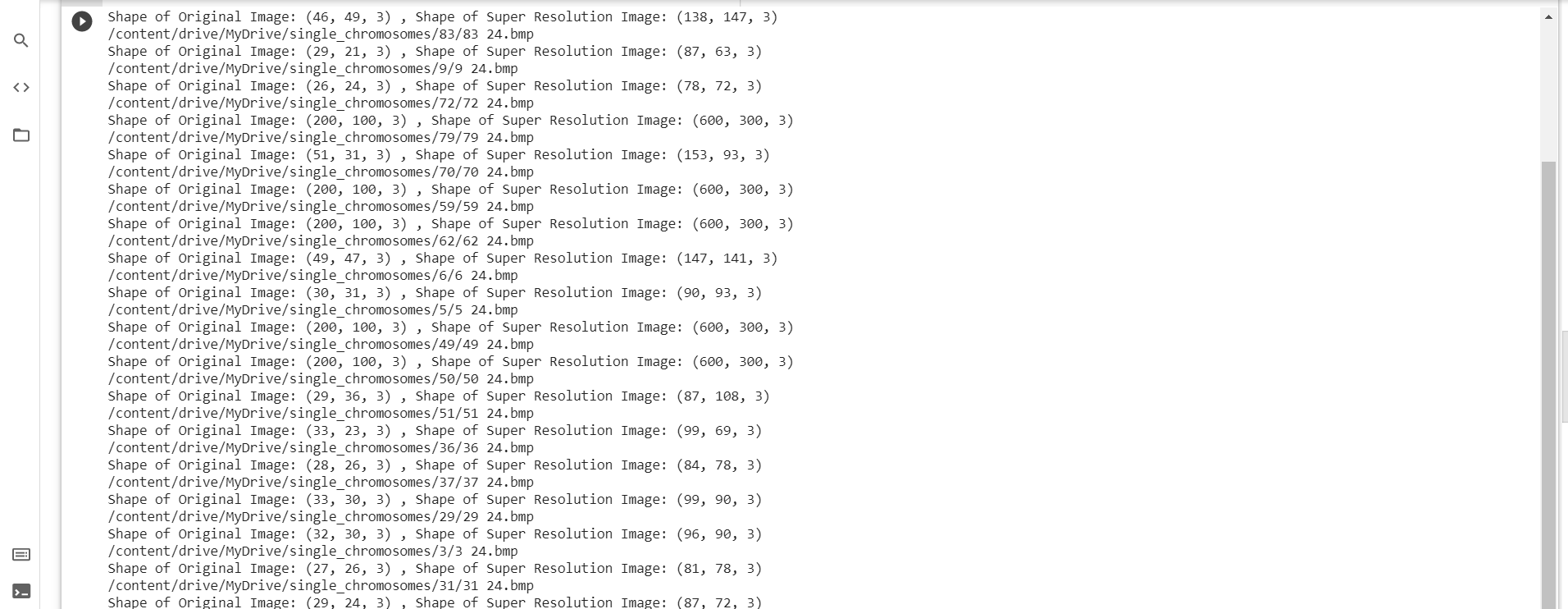
**4.2.5.2. IMAGE RECONSTRUCTION**

At level *s*, the input image is upsampled by a scale of 2 with a transposed convolutional (upsampling) layer. This layer is initialized with the bilinear kernel.

The upsampled image is then combined (using element-wise summation) with the predicted residual image from the feature extraction branch to produce a high-resolution output image.

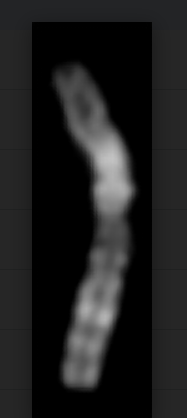
The output HR image at level *s* is then fed into the image reconstruction branch of level *s*+1.

We have used Laplacian Pyramid Super Resolution Network because it outperforms all the other super resolution models present. This network had enhanced the resolution of our Chromosome image and thus helped in yielding better accuracy.



(Fig 4.2.5 Output of Laplacian Pyramidal Super Resolution)

The below images are each strand of the Chromosome which was segmented after Identification and Segmentation through our Detectron2 object detection model. The resolution of the left image (20x20) and the resolution of the right image (120x120) are shown below:



(Fig 4.2.6 Image Before Super Resolution) (Fig 4.2.7 Image After Super Resolution)

**CHAPTER 5**

**PROPOSED METHOD**

**5.1 COMPUTATION AND TRAINING TOOLS**

Two tools were used for training of the classification model. These two tools were :

1. GPU - hardware based for computing
2. Colab - cloud based for computing

**5.1.1. GPU:**

A GPU (Graphics Processing Unit) is a specialized processor with dedicated memory that conventionally perform floating point operations required for rendering graphics.

In other words, it is a single-chip processor used for extensive Graphical and Mathematical computations which frees up CPU cycles for other jobs.

The main difference between GPUs and CPUs is that GPUs devote proportionally more transistors to arithmetic logic units and fewer to caches and flow control as compared to CPUs.

While CPUs are mostly applicable for problems that require parsing through or interpreting complex logic in code, GPUs are designed to the dedicated graphical rendering workhorses of computer games, and which were later enhanced to accelerate other geometric calculations (for instance, transforming polygons or rotating verticals into different coordinate systems like 3D).

A GPU is smaller than a CPU but tends to have more logical cores (arithmetic logic units or ALUs, control units and memory cache) than the latter.

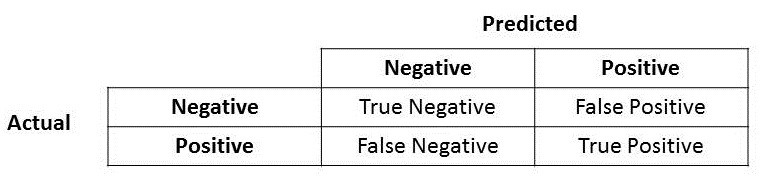


(Fig 3.1.1 GPU)

**5.1.2. COLAB:**

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

**5.2. METRICS:**



Precision is a good measure to determine when the cost of False Positive is high. For instance, email spam detection. In email spam detection, a false positive means that an email that is non-spam (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model.

Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive). Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.

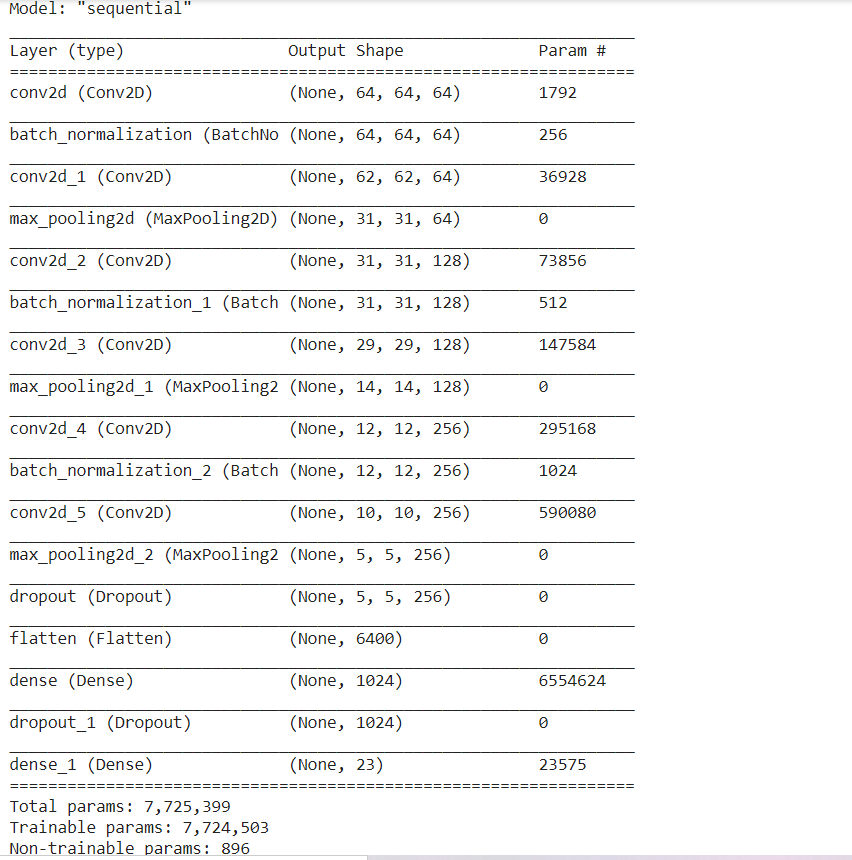
F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives).

**5.3. PROPOSED MODELS**

**5.3.1. CUSTOM MODEL**



(Fig 5.1.1 CNN Layer)



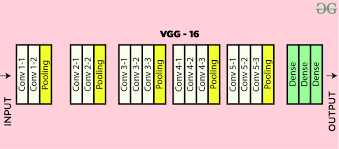
(Fig 5.1.2 Sequential Model for CNN Architecture)

**5.3.2. RESNETS**



(Fig 5.2.1 ResNet Architecture)

**5.3.3. VGG Net**

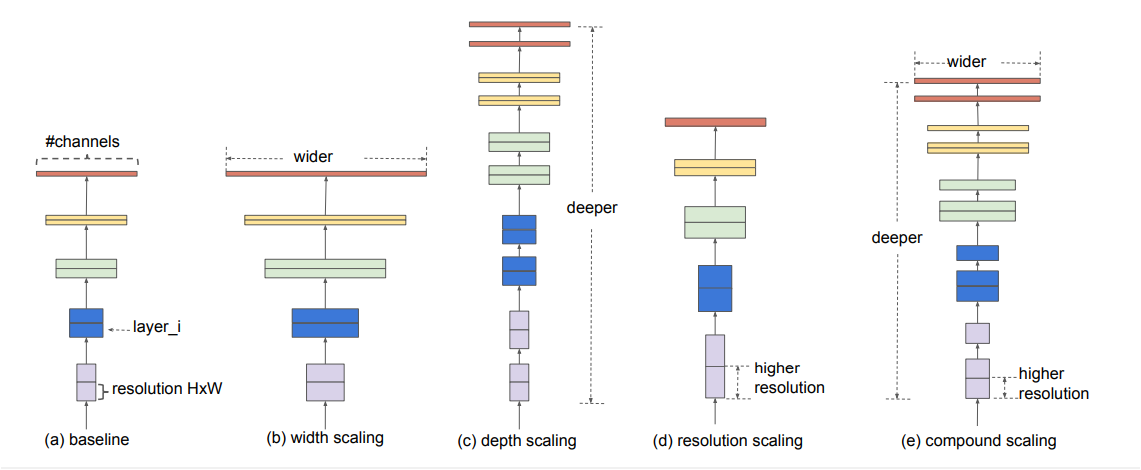


(Fig 5.3.1 VGG - NET Architecture)

**5.3.4. EFFICIENT NETS**

**5.3.4.1. OVERVIEW**

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a *compound coefficient*. Unlike conventional practice that arbitrarily scales these factors, the EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients.

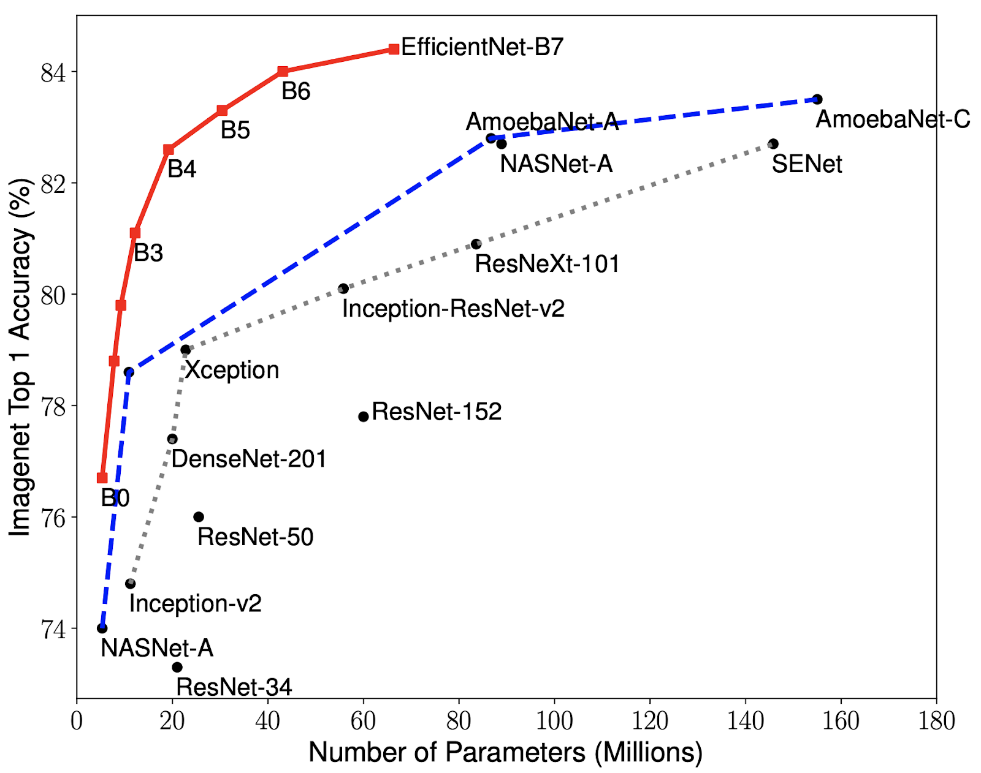


(Fig 5.4.1 Different scaling methods for EfficientNet )

Model Scaling. (a) is a baseline network example;

(b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution.

(e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.



(Fig 5.4.2 EfficientNets Vs ResNet )

As we can learn from the graph that :

* The highest accuracy is achieved with EfficientNet-B7.
* There's less number of parameters used when compared with other models.

**5.3.4.2. SWISH ACTIVATION**

ReLu works pretty well but it has a problem, it nullifies negative values and thus derivatives are zero for all negative values. There are many known alternatives to tackle this problem like leaky ReLu, Elu, Selu etc., but none of them has proven consistent.

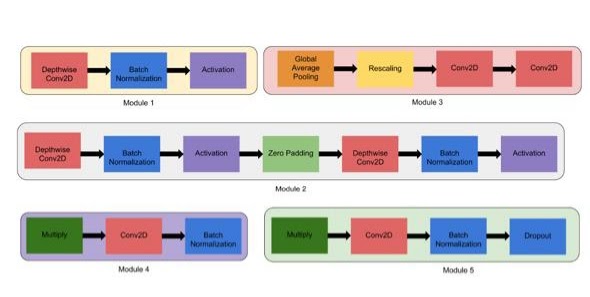
The Google Brain team suggested a newer activation that tends to work better for deeper networks than ReLU which is a Swish activation. They proved that if we replace Swish with ReLu on InceptionResNetV2, we can achieve 0.6% more accuracy on the ImageNet dataset.

Swish is a multiplication of a linear and a sigmoid activation.

Swish(x) = x \* sigmoid(x)

**5.3.4.3. BASIC STRUCTURE:**

The total number of layers in EfficientNet-B0 is 237 and in EfficientNet-B7 the total comes out to 813. So to make it easy to comprehend, these layers can be made from 5 modules shown below.



(Fig 5.4.3 Base Structure of Efficient Nets)

* **Module 1** — This is used as a starting point for the sub-blocks.
* **Module 2** — This is used as a starting point for the first sub-block of all the 7 main blocks except the 1st one.
* **Module 3** — This is connected as a skip connection to all the sub-blocks.
* **Module 4** — This is used for combining the skip connection in the first sub-blocks.
* **Module 5** — Each sub-block is connected to its previous sub-block in a skip connection and they are combined using this module.

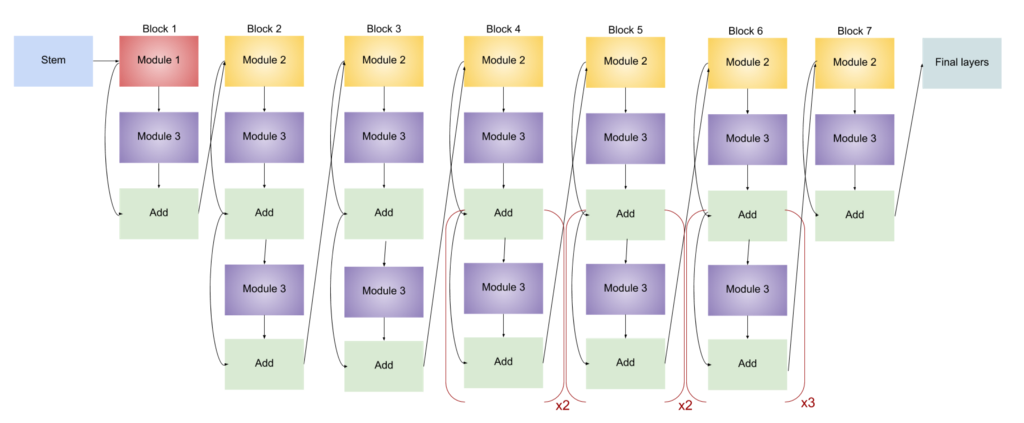
These modules are further combined to form sub-blocks which will be used in a certain way in the blocks.

We have worked with 3 models of EfficientNet:

* B0
* B5
* B6

**5.3.4.4 B0 Model:**

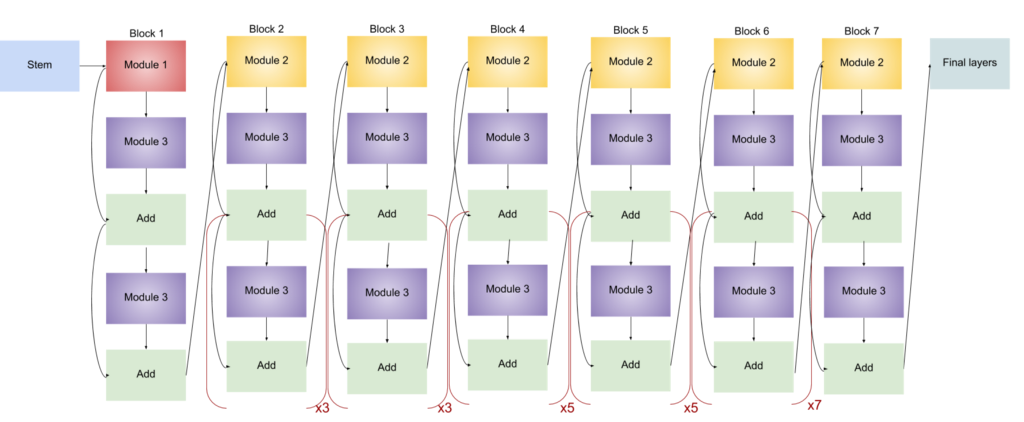
There are 237 layers in EfficientModel-B0.This act as a baseline model for the remaining EfficientNets model.



(Fig 5.4.4 Architecture of EfficientNet B0 model)

**5.3.4.5 B5 Model:**

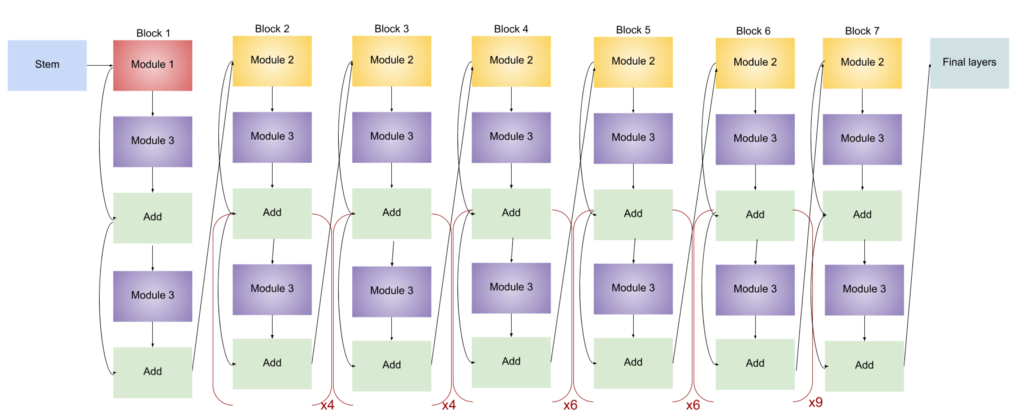
There are 521 layers in EfficientNet B5 model.



(Fig 5.4.5 Architecture of EfficientNet B5 model)

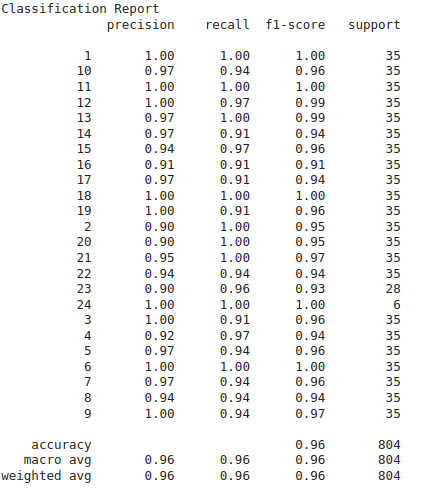
**5.3.4.6. B6 Model:**

There are 671 layers in EfficientNet B6 model.



(Fig 5.4.6 Architecture of EfficientNet B6 model)

**5.3.5. CLASSIFICATION REPORT ON B6 + Custom last layers MODEL:**



(Table 5.4.1 Classification report - B6 model)

**5.3.6. MODEL SUMMARY :**

| Model | Accuracy |
| --- | --- |
| Efficient Net B0 (Without Super Resolution) | 88.75 |
| Efficient Net B0 | 92.06 |
| Efficient Net B5 | 95.2 |
| Efficient Net B6 | 96.6 |
| VGG16 | 94 |
| Custom Model | 88 |

(Table 5.4.2 Model Summary)

From the above table , it can be inferred that the highest accuracy is achieved using EfficientNet B6.

The difference between the accuracy of the EfficientNet B6 model and Custom Model is notable.

**5.3.7. Accuracy calculation formula and calculation :**

* No of correct predictions/ Total No. of predictions
* Total No of predictions = 804
* No. of correct predictions = 777
* Therefore, Accuracy= 1930/2000 = 0.966
* This is 96.6% accuracy for our EfficientNet B6 model

**5.3.8. COMPARISON**:

| Model | Accuracy | Precision | Recall | F1 Score |
| --- | --- | --- | --- | --- |
| Previous State of The Art | 94.04 | 94.51 | 93.86 | 94.18 |
| Our Approach | 96.60 | 97.01 | 96.1 | 96.41 |

(Table 5.4.3 Comparison with other state of the art models )

**CHAPTER 6**

**FUTURE SCOPE**

**6.1. ABNORMALITY DETECTION**

Chromosomes carry our individual genetic material (genes). Our bodies are made up of billions of cells, and in each cell there are 23 pairs of chromosomes. Extra or missing chromosomes may cause abnormalities that can affect a baby's development. The most common chromosome abnormality affecting newborns is Down syndrome which occurs in around 1 in 500 to 600 pregnancies.

**6.1.1. NUMERICAL ABNORMALITIES**

An abnormal number of chromosomes is called aneuploidy, and occurs when an individual is either missing a chromosome from a pair (resulting in monosomy) or has more than two chromosomes of a pair (trisomy, tetrasomy, etc.).Aneuploidy can be full, involving a whole chromosome missing or added, or partial, where only part of a chromosome is missing or added.An example of trisomy in humans is Down syndrome, which is a developmental disorder caused by an extra copy of chromosome 21; the disorder is therefore also called trisomy 21.

An example of monosomy in humans is Turner syndrome, where the individual is born with only one sex chromosome, an X.

**6.1.2. STRUCTURAL ABNORMALITIES**

When the chromosome's structure is altered, this can take several forms:

* Del: A portion of the chromosome is missing or has been deleted. Known disorders in humans include [Wolf-Hirschhorn syndrome](https://en.wikipedia.org/wiki/Wolf-Hirschhorn_syndrome), which is caused by partial deletion of the short arm of chromosome 4; and [Jacobsen syndrome](https://en.wikipedia.org/wiki/Jacobsen_syndrome), also called the terminal 11q deletion disorder.
* Duplications: A portion of the chromosome has been duplicated, resulting in extra genetic material. Known human disorders include [Charcot-Marie-Tooth disease type 1A](https://en.wikipedia.org/wiki/Charcot-Marie-Tooth#Types_of_the_disease), which may be caused by duplication of the gene encoding [peripheral myelin protein 22](https://en.wikipedia.org/wiki/Peripheral_myelin_protein_22) (PMP22) on chromosome 17.
* Inversions: A portion of the chromosome has broken off, turned upside down, and reattached, therefore the genetic material is inverted.
* Insertions: A portion of one chromosome has been deleted from its normal place and inserted into another chromosome.
* Translocations: A portion of one chromosome has been transferred to another chromosome. There are two main types of translocations:
  + [Reciprocal translocation](https://en.wikipedia.org/wiki/Chromosomal_translocation#Reciprocal_translocations): Segments from two different chromosomes have been exchanged.
  + [Robertsonian translocation](https://en.wikipedia.org/wiki/Robertsonian_translocation): An entire chromosome has attached to another at the [centromere](https://en.wikipedia.org/wiki/Centromere) - in humans these only occur with chromosomes 13, 14, 15, 21, and 22.
* Rings: A portion of a chromosome has broken off and formed a circle or ring. This can happen with or without loss of genetic material.
* Isochromosome: Formed by the mirror image copy of a chromosome segment including the centromere.

Chromosome instability syndromes are a group of disorders characterized by chromosomal instability and breakage. They often lead to an increased tendency to develop certain types of malignancies.

**CHAPTER 7**

**CONCLUSION**

The main objective of the project was to create a state of the art classification model to classify the chromosome. This project represents work with various deep learning models such as ResNets, VGG-16, a very own custom model and EfficientNets (B0,B5,B6). After training the model, it was found that this model received the best accuracy while using the EfficientNet B6 model. An accuracy of 96.66% was achieved. When researching with the classification models currently available to classify the chromosome, the best accuracy in the market was achieved as well.

Apart from classifying, the detection of chromosomes was possible by using two object detection models, YoloV4 and Detectron2. Using YOLOV4, the model was not able to detect each strand of chromosome but through Detectron2 model, the model was able to detect each strand of the chromosome. These strands were used as training and testing sets for the classification model.

The main reason for such a high accuracy was through Super Resolution. When the non super resolution image was given as data for the classification model, it yielded accuracy of 1-2% lower than the ones with the super resolution image.

After collecting sufficient data, the detection of any abnormalities in the chromosomes is possible. This will in turn help the doctors in speeding the karotyping process and finding any irregularities.

**REFERENCES:**

* M. J. Roshtkhari and S. K. Setarehdan, “A novel algorithm for straightening highly curved images of human chromosome,” Pattern Recognition Letters, vol. 29, no. 9, pp. 1208–1217, 2008.
* Chromosome Classiﬁcation with Convolutional Neural Network based Deep Learning.2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)
* MONA SALEM AL-KHARRAZ - July 2020 -Automated System for Chromosome Karyotyping to Recognize the Most Common Numerical Abnormalities Using Deep Learning.
* X. Wang, B. Zheng, S. Li, J. J. Mulvihill, M. C. Wood, and H. Liu, “Automated classification of metaphase chromosomes: optimization of an adaptive computerized scheme,” Journal of Biomedical Informatics, vol. 42, no. 1, pp. 22–31, 2009
* Swati, Monika Sharma, and Lovekesh Vig - 2018 Automatic Classification of Low-Resolution Chromosomal Images.
* Swati, G. Gupta, M. Yadav, M. Sharma, and L. Vig, “Siamese networks for chromosome classification,” in Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 72–81, IEEE, Venice, Italy, October 2017
* Swati, Monika Sharma, and Lovekesh Vig - 2018 Automatic Chromosome Classification using Deep Attention Based Sequence Learning of Chromosome Bands.
* F. Abid and L. Hamami, “A survey of neural network based automated systems for human chromosome classification,” Artificial Intelligence Review, vol. 49, no. 1, pp. 41–56, 2018.
* M. Sharma, O. Saha, A. Sriraman, R. Hebbalaguppe, L. Vig, and S. Karande, “Crowdsourcing for chromosome segmentation and deep classification,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 786–793, IEEE, Honolulu, HI, USA, July 2017
* L. Ji, “Fully automatic chromosome segmentation,” Cytometry, vol. 17, no. 3, pp. 196–208, 1994.
* S. Minaee, M. Fotouhi, and B. Khalaj, “A geometric approach to fully automatic chromosome segmentation,” in Proceedings of the Signal Processing in Medicine and Biology Symposium, pp. 1–6, IEEE, Philadelphia, PA, USA, December 2014.
* V. S. Balaji and S. Vidhya, “Separation of touching and overlapped human chromosome images,” in Advancements of Medical Electronics, pp. 59–65, Springer, Berlin, Germany, 2015.