Topik : Image Processing for Tuberculosis Detection in Sputum Sample

Nomor Kelompok : 15

Kelas : LK01

Ketua : 2602077553 - Alvin Linardi

Anggota : 2602061353 - Kanaya Ravensca Childira

**Kontribusi dalam project:**

|  |  |  |  |
| --- | --- | --- | --- |
| NIM | Nama Lengkap | Kontribusi | Keterangan |
| 2602077553 | Alvin Linardi | 1. Melakukan Review Artikel Sejumlah 10 | Artikel yang saya review dapat dilihat dari link berikut dengan tabel yang berwarna biru:  <https://docs.google.com/spreadsheets/d/1tPDfnrgRBkI_0xW2exqnZj7ECn8t9C1aqYqX5eWfp3o/edit?usp=sharing> |
| 1. Membuat Desain Workflow | Desain Workflow yang telah saya kerjakan dapat dilihat melalui link :  <https://drive.google.com/file/d/1xHq8zdTdct7gULWPWmFIl_yfwC3CafNB/view> |
| 1. Merancang Code Klasifikasi menggunakan Model FasterRCNN\_Resnet50\_FPN, EfficientNetV2, Inception dan VGG16 | Code ini terinspirasi /bersumber dari <https://www.kaggle.com/code/ashreyareddy/tb-95-2> dan terdapat modifikasi yaitu di duplikat menjadi 3 code lagi yang telah terdapat modifikasi pada bagian penggunaan model yang terdapat pada code yang diganti menjadi EfficientNetV2, Inception, dan VGG16 |
| 1. Memperbaiki Code dari saya sendiri yang mana memanfaatkan arsitektur EfficientNetV2, Inception, dan VGG16 yang kemudian diganti menjadi varian lain dari FasterRCNN\_Resnet50\_FPN yaitu FasterRCNN\_Resnet50V2\_FPN, FasterRCNN\_MobileNetV3\_FPN dan FasterRCNN\_MobileNetV3\_large\_FPN | Ternyata error sebelumnya disebabkan oleh penggunaan EfficientNetV2, Inception, dan VGG16 yang saya coba jadikan sebagai backbone dari FasterRCNN yang ternyata tidak bisa dan menimbulkan error sehingga terdapat penyesuaian pada pemilihan model dimana 3 model lainnya diganti menjadi model-model yang menjadi varian-varian lain dari FasterRCNN tersebut. |
| 2602061353 | Kanaya Ravensca Childira | Melakukan Review Artikel Sejumlah 10 | Artikel yang saya review dapat dilihat dari link berikut dengan tabel yang berwarna ungu:  <https://docs.google.com/spreadsheets/d/1tPDfnrgRBkI_0xW2exqnZj7ECn8t9C1aqYqX5eWfp3o/edit?usp=sharing> |
| Penulisan Introduction (Progress 2) | Menjelaskan konteks dan latar belakang masalah yang sedang dibahas, kondisi saat ini, serta memberikan penjelasan mengapa solusi yang diusulkan merupakan pilihan yang baik dan optimal. |
| Pembuatan PPT Presentasi | Membuat presentasi PowerPoint menggunakan Canva yang dapat menjelaskan keseluruhan proyek dari latar belakang hingga hasil. |
| Penulisan Introduction (Laporan) | Memperluas dan mengerjakan ulang pendahuluan asli dari progres 2 untuk menyesuaikan dengan template yang telah disediakan. |
| Paper Formatting | Memformat laporan untuk penampilan yang lebih baik dengan:  1. Mengubah jenis dan ukuran huruf  2. Membuat indentation pada awal setiap paragraf  3. Menggunakan huruf tebal untuk menyoroti tema utama  4. Menambahkan komentar  5. Memanfaatkan aplikasi Mendeley  6. Menambah caption pada setiap gambar yang terlampir dalam laporan ini |

**IMAGE PROCESSING FOR TUBERCULOSIS DETECTION IN SPUTUM SAMPLE**

**Introduction**

Tuberculosis (TB) is an infectious respiratory illness caused by the bacterium called Mycobacterium tuberculosis (Tobin & Tristram, 2024). It primarily affects the lungs, though it can also target other parts of the body such as the kidneys, spine, or brain. When a person infected with tuberculosis coughs, sneezes, or spit, they transmit the bacteria via the air thus putting those nearby at risk. Tuberculosis is both preventable and curable, antibiotics being the primary treatment of choice. The Bacille Calmette-Guérin (BCG) vaccination was developed in 1921 to tackle this among babies and small children in certain countries though the vaccine only prevents TB outside of the lungs, not inside.

Tuberculosis symptoms might include a persistent cough with or without blood, as well as discomfort in the chest, weakness, fatigue, a loss of appetite, a high body temperature, and sweating throughout the night(Okafor et al., 2024). A compromised immune system like HIV and AIDS, diabetes, malnutrition, and the use of tobacco products all increase the risk of acquiring tuberculosis (Chiang et al., 2023). Early identification is critical for preventing the spread of tuberculosis, especially because symptoms may be minor at first but eventually leading to an acceleration of symptoms. To avoid tuberculosis infection and transmission, people should seek medical attention as soon as they notice symptoms, be tested if they are at risk, finish prescribed treatments, and practice excellent hygiene, including proper manners regarding coughing and sputum removal (Lovering, 2023). These preventive strategies are critical in slowing the development of tuberculosis and mitigating its effects on public health.

Despite its preventability however, it should be known that around one-quarter of the world’s population has already been infected by the Mycobacterium tuberculosis bacterium, while 5-10% would start experiencing symptoms and then eventually develop the tuberculosis disease (World Health Organization, 2023b). In 2022 alone, tuberculosis continued to be a major global health issue as it claimed the lives of over 1.3 million people, including 167.000 of those also affected by HIV. This makes tuberculosis the second leading cause of mortality after COVID-19, outranking HIV/AID on cause of deaths. In the same year, 10.6 million people was infected with tuberculosis, impacting men, women, and children worldwide in various countries from various age demographics.

Without proper treatment for it, the death rate for tuberculosis is high, reaching about 50%. However, with prevention measures, the cure success rate is around 85%. Despite efforts from global representatives, tuberculosis-related mortality has only decreased by 19% from 2015 to 2022, falling short from the reduction goal of 75% by 2025(World Health Organization, 2023) . This shortfall of reduction spotlights the critical need for ongoing attention and action. Nonetheless, it should be also noted that about 75 million lives were saved since the year 2000 due to continuous global initiatives.

One method to diagnose someone with tuberculosis by doing a sputum test. Not to be confused with the saliva, sputum is the phlegm that comes from deep in the lungs and that is where TB bacteria could be detected. A sputum test is carried out like this: the patient is instructed to cough deeply to expel sputum from their lower respiratory tract then a healthcare provider will provide a sterile container for collecting the sample. After the collection, the container is carefully sealed to avoid contamination with outside factors. It is then delivered to the lab for testing. In the laboratory, the sputum sample is subjected to a variety of tests to determine the presence of Mycobacterium tuberculosis. These tests may include smear analysis using a microscope, in which the sample is stained and studied by lab experts for the presence of TB bacteria or growing a sputum culture where they allow TB bacteria to proliferate and be easily identified further. A sputum test may serve as a good detection tool for early case finding, as it can detect the Mycobacterium tuberculosis germs, so that health care providers may diagnose the disease, even prior to the showing of any symptoms on the patient. This warrants early treatment and recovery, while it lessens the dangerous risk of it spreading to others. The timeframe for the results of a manual sputum tuberculosis test is quite variable. The result can be gathered within a few days to even several weeks. This test is generally deemed to be tedious and inefficient, especially for sputum culture analysis. Even the process may be very subjective in nature and quite variable among different observers. Different people might interpret the same sample differently while looking at it under a microscope, therefore giving different results, which may lead to wrong diagnosis as well (Asgharzadeh et al., 2020).

Besides, sputum tests demand specialized equipment, and this fact may be a real challenge in resource-limited areas—where funding for buying, maintaining, or installing required equipment may be poor, as well as the poorest distribution network for such equipment (Kabir et al., 2021). Therefore, this system needs to have some degree of automation, as is discussed following. Machine learning with image processing indeed is a toolbox for attacking the issues. These machine learning algorithms may standardize sputum samples in an objective way, hence decreasing variability and increasing reliability in such a way that the problem of mixed interpretations from different people will be reduced (Ghaffar Nia et al., 2023). Moreover, the use both image processing and machine learning in one system can drastically cut the time for results, making it not laborious but rather effective in terms of time. Finally, its software may be installed on low-costing computing platforms so as to make advanced diagnostics more accessible in resource-constrained areas (Pilgrim & Prabhakara, 2021).

A systematic literature review was carried out to expand the authors horizons of the various methods that were used to create an automated tuberculosis detection model. Using Google Scholar as the primary search engine for this review, we explored multiple academic literatures on this specific topic of image processing for tuberculosis detection. The academic paper used were written in English and must have been published after 2020, failure to comply results in the paper not being used for our review.

The first study reviewed focused on improving the efficiency of TB detection by automating the counting processes of TB bacteria in sputum images. The dataset used were 100 digital images of sputum samples from TB patients that’s divided into 75 in-samples/training images and 25 out-samples/testing images. The method used in this study uses image processing to ensure the quality of the images used, dimension reduction using Discrete Wavelet Transform (DWT) and Partial Least Squares (PLS) to reduce the number of predictor variables from 2048 variables to only 5, and application of non-parametric and parametric regression models to count the estimate number of TB bacteria. Discrete Wavelet Transform is a transform that breaks down a specific signal into several sets, each seat being a time series of parameters defining the time evolution of the signal in the associated frequency range(Hosseinzadeh, 2020). Partial Least Squares (PLS) refers to a statistical technique using several response variables supposedly related to more than one explanatory factor. PLS is particularly relevant for multi-regression set-ups under small sample sizes, missing data, or multicollinearity issues, given that this is a technique on the basis of covariances (Xia, 2020).

In another study, the non-parametric Poisson regression model that used the local linear estimator resulted with better accuracy and speed in the output. It recorded an accuracy of 82.75%, with its response time active compared to the parametric linear regression model, which recorded an accuracy of 95.5% and took a long time to give the response of the result. The deviance value for the non-parametric approach has reduced compared to the parametric approach, showing the score of 28.410, whereby the parametric approach scored 93.029. Concluding that the non-parametric approach is much more efficient for counting TB bacteria in sputum samples.

Another study reviewed focused on improving the accuracy of detecting TB by utilizing Convolutional Neural Networks (CNN) to analyze chest X-ray images. CNN is a deep learning algorithm that is most commonly used in image processing (Umam & Handoko, 2021). CNN is made up of multiple layers such and is inspired by the visual process of the human brain, making it perfect for pattern hierarchy detection yet heavily depends on the images spatial. The dataset used were 406 images of healthy lungs chest X-rays and 394 abnormal lungs, which then later would be expanded with an additional 239 health and 554 abnormal lungs. The methods used were first to test the impact of an image’s resolution towards the end results, the usage of pre-trained networks for transfer learning purposes, adjustment of some parameters and functions then data augmentation to optimize CNN performance. Data augmentation refers to a strategy of artificially increasing the size and variety of a dataset without having to look for more samples. It is done by adding slightly modified duplicates of the original image using techniques such as rotating images, scaling (the resizing of the images), translations (flipping the image horizontally or vertically), cropping (zooming-in on some images) and noise implementations(Khan et al., 2023).

The study found that the optimal image size is 64x64 due to having it in a higher resolution may result in overfitting, the pre-trained model showed reasonable, but not satisfactory results due to their initial training learning from non-medical images, the parameter and function adjustments did not cause any improvement in accuracy, however, data augmentation showed the best results as it significantly enhanced the TB detection. In conclusion, data augmentation is crucial for CNN performance optimization (Norval et al., 2021).

A study was also conducted to compare various machine learning algorithms performance on detecting tuberculosis from chest X-ray images. The dataset used is the Montogomery database from the UNDP tuberculosis control program from Iraq that contains 138 X-ray images where 80 images are healthy and 58 are infected with tuberculosis. The machine learning algorithms in application here were Support Vector Machines, Logistic Regression, and K-Nearest Neighbors (KNN). Support Vector Machine (SVM) is a powerful algorithm in supervised learning for classification and regression. SVM works by finding a separating hyperplane in the data space that maximizes the margin of data points from different classes.

In a recent conducted study, SVM showed extremely good performance, with 87.9% in accuracy, 85.9% in precision, 85.9% in recall, and 84% in F1 against logistic regression, having 80.8% in the accuracy field, 81.8% in precision, 80.8% in recall, and 79% in the F1 field. All these achieved rates place logistic regression between the acquired 80.8% in accuracy, 81.8% in precision, 80.8% in recall, 79% in F1, while K-NN showed the least performance, with 75.8% in accuracy, 77.8% in precision, and an F1 score of 74%. Hence, SVM performed better than logistic regression and K-NN, thereby making it suitable for the detection of tuberculosis based on these metrics (Alsaffar et al., 2021).

The proposed current work is devoted to the development of an automated system using Convolutional Neural Network algorithms for the early detection of tuberculosis from sputum samples. CNN is a deep learning technique, largely influential in image processing, mainly with pre-trained models. A pre-trained model that will be used for this project comes from the torchvision library. One that will be used is the FasterRCNN\_Resnet50\_FPN, which is based on the popular COCO dataset. This study uses the Tuberculosis image dataset 19d4ae81-a, which was collected from different sources. It consists of 2530 sputum sample images, which hold characteristics related to TB diagnosis. It is anticipated that this development may achieve high precision, recall, and accuracy in identifying TB cases. The automated manner with which to detect TB from sputum samples makes the system accurate in its tests, is much faster in terms of turn-around time, less dependent on manual analysis, and requires less computational resources, which will thereby enable more accessibility in resource-constrained settings.

**METHODOLOGY**

1. **Data Understanding**

The Tuberculosis Image Dataset is a collection of 928 images that represent sputum samples used in diagnosing tuberculosis disease. A total of 3,734 bacilli were annotated with bounding boxes. The dataset consists of images in the format of .jpg, corresponding to the format of .xml files enlisting the details of bounding boxes obtained by such specialized machinery to aid the training of machine learning models for TB detection. In this regard, the data set is of much importance, adding to the research work in developmental automated TB diagnostics. Bounding boxes will thus help the machine learning algorithms to detect TB bacilli in sputum samples. This feature takes big importance in detailed and high-quality annotations for ensuring efficient model training and model performance in real-world diagnostic implementations. This dataset is therefore instrumental because bounding-box data are available on the images in the development and implementation of tuberculosis diagnostic tools. It further provides the necessary data for the development and enhancement of TB detection technology and for this reason, an invaluable piece of data in the fight against tuberculosis. (Kaggle, n.d.).

A screenshot of a graph

Description automatically generated

Fig 2.1 Dataset information on the image specification and the bounding boxes existed in the dataset



Fig 2.2 Image example from the dataset equipped with the bounding boxespap

1. **Data Exploration**

The research must begin by defining the number of classes that shall be classified. Before that, the dataset must be explored thoroughly to further understand the structure of the dataset itself. The dataset consists of 928 images of sputum samples. Across the 928 images, there are 3734 bacilli instances bound in a red-colored bounding box as a mark of the presence of the bacilli. The bounding box is stored in a .xml file for each image. After further understanding of the data, a certain amount of classes can be determined for this research. Two classes will be classified for this research for identification, which consist of TB-Positive class and TB-Negative class.

1. **Data Splitting**

This process aims to split the dataset into a certain portion to train and validate the dataset. The result of this process will be a trained dataset. This will help fine-tune the pre-trained model of FasterRCNN\_ResNet50\_FPN used later on. The splitting ratio values at 0.2, representing an 80% Training set and a 20% Validation/Testing set.

This process is done using the scikit-learn model to create the training and the validation/testing model. The training and validation/testing model will later be used to fit the pre-trained model of the CNN model architecture. To do this process, variables of training and testing/validation dataset are created (e.g., train\_ds, val\_ds, train\_dl, val\_dl). These variables will be used to store the split dataset assigned for each task, either training or testing. After this process is done, model selection will be required.

1. **Model Selection**

For this research, FasterRCNN\_Resnet50\_FPN is the main model to be used, which is acquired from torchvision detection model libraries. This model will be fitted to the previously made training and testing/validation dataset. The pre-trained model will learn from the training and testing/validation dataset to detect the existence of TB Bacteria, which will then be determined in two classes, TB-Positive and TB-Negative, based on the sputum sample image, which will then be used as a part of the detection testing.

Faster R-CNN (Region-based Convolutional Neural Networks) with ResNet50 backbone and Feature Pyramid Networks (FPN) is a state-of-the-art object detection model. The Faster R-CNN framework consists of two stages: the first stage generates region proposals, and the second stage classifies these proposals and refines their bounding boxes. ResNet50, a deep convolutional network with 50 layers, is utilized for its exceptional ability to extract detailed visual features. The inclusion of the Feature Pyramid Network (FPN) enhances detection accuracy by creating feature pyramids that maintain high-level semantic information across all scales (Lin et al., 2017).

A screenshot of a computer program

Description automatically generated

Fig 1.3 Model Building FasterRCNN\_ResNet50\_FPN

**The FasterRCNN\_ResNet50\_FPN** architecture combines a Region Proposal Network (RPN) with ResNet50. The RPN suggests regions of interest (RoIs), which are then further refined in the network's second stage. The FPN improves the backbone by merging low-resolution, semantically rich features with high-resolution, semantically weaker features, enhancing the detection of objects at various scales (Ren et al., 2015).The equations governing the RPN include the objectness score and bounding box regression:

The variable represents the predicted probability that anchor 𝑖 contains an object, while is the actual ground-truth label.The terms and denote the predicted and ground-truth bounding box coordinates, respectively. The scailing of these two loss components is balanced by the parameter λ.

The numeric specifications below provide some detail on what goes on at the back of the implemented Faster R-CNN model which included :

**Normalization and Resizing**: The input to the images is normalized using values of mean [0.485, 0.456, 0.406] and standard deviations [0.229, 0.224, 0.225], not different from several other common-place pre-processing settings for deep learning models on image data. The resizing of input images is scaled using a dynamic scale of minimum size with bilinear interpolation.

**Convolutional Layers**: It builds the backbone with many convolutional layers and sets out with a 7 × 7 convolution with 64 filters, stride 2, and padding 3. It sets the bare footing for numerous steps, which heavily use bottleneck blocks. Each bottleneck configuration in the ResNet layers mainly consists of 1×1 and 3×3 kernel convolutions, where the count of filters and strides has been selected thoughtfully so that the size of the feature maps along the depth is maintained.

**FPN (Feature Pyramid Network)**: The entire construction of the FPN guarantees, with every level of transition, a 1×1 convolution takes place in such a manner that the quantity of channels held constant throughout the pyramid is 256. This ensures the proper mixing of features at different levels.

**Region Proposal Network (RPN):** is followed by a 3x3 convolutional layer with 256 filters—classification and Bounding box regression heads as well as 256-channels, but they have 1x1 kernel size. The snippet does not explain in detail the anchor generator parts. Still, in general, these come with features such as the scale and the aspect ratios of the anchors, which are crucial in the bounding boxes' initialization.

**Heads of RoI**: The output from the RoI pooling layer is set to be of a fixed size 7x7 given RoI of any size; hence, this helps in focusing on the feature extraction that is residing inside the RoI. The two-layer MLP head does bounding box refinement and classification. Feature transformation is drastic in the output of the pooling layer in that, from the 12,544 input features, the transformation is reduced to 1024 in both fully connected layers.

**Output Sizes**: The final classification and bounding-box regression layers of the Fast R-CNN predictor are task-specific, e.g., :

• a classification head makes scores for two classes

• a box head makes predictions for eight values that probably are correction values to 4 coordinates over classes..

**FasterRCNN\_ResNet50\_FPN\_V**2 is a much-refined model building on the previous research of Faster R-CNN. This work has a few key optimizations over its model to boost performance. The major ones are advanced normalization techniques like Group Normalization and more precise strategies in the anchor generation; these updates are carried out to bring better stability in training and enhance the capacity of the network to be able to handle objects of various scales and different aspect ratios (Wu & He, 2018).

The backbone was fixed on ResNet50, but on both the stages of FPN and RPN, improvements were added to increase multiscale feature representation with region proposals accuracy. The revised method of anchor generation using a bigger range of aspect ratios and scales allows for more accurate region proposals, making it possible for the network to localize objects of different shapes and sizes more clearly. In this way, the accuracy of region proposals increases (Lin et al., 2017). More details on the FasterRCNN\_ResNet50\_FPN\_V2 included :

**GeneralizedRCNNTransform :**

* Normalize: The image will be normalized with the general mean values, as [0.485, 0.456, 0.406], and the standard deviation, as [0.229, 0.224, 0.225]. These are usual normalizing values coming from models pre-trained on ImageNet.
* Resize: It resizes the input images with a scaling transformation happening while preprocessing the image, with its minimum size being 800 pixels and a maximum size of 1333 pixels. Since resizing is considered the most critical operation, it can consume or take away from input quality representation, so it often becomes very critical to do it carefully so that quality is not lost much.
* Backbone: The backbone with FPN (Feature Pyramid Network) layers is built as a sequential structure of bottleneck blocks, each shared with layers such as Conv2d and BatchNorm2d.

These describe concrete bottlenecks as an increase in channels through layers:

* This first layer: Channel size 64 to 256.
* Second layer: Channel size 256 to 512.
* Third layer: Channel size 512 to 1024.
* Fourth layer: Channel size 1024 to 2048.

Parts of an FPN: Builds a multi-scale pyramid of an input image: each level with resolution decreases by a factor of 2, and channels are 256 using Conv2dNormActivation.

**RPN (Region Proposal Network)**

* Anchor Generator: In other words, it's the anchor generator region proposed generator at different scales and aspect ratios.
* Head: It is built by the chain of convolution layers in 256 channels followed by the ReLU activation. It gives two outputs:
* cls\_logits: 3 output channels foreground, background, and another class for classifying anchor as an object.
* bbox\_pred: 12 output channels predicting four coordinate adjustments of three anchors per location.

**ROIHeads**

* MultiScaleRoIAlign: It aligns the feature maps of all scales and returns feature maps of a fixed size, which is 7x7.

**FastRCNNConvFCHead:**

* Conv layers: layers of convolutions, several layers, all with 256 channels, and the same spatial dimensions.
* Flatten: flattening the output from the convolutional layers.
* Linear layers: reduce the features to 1024 by applying the flattened output to a linear layer.

**FastRCNNPredictor**:

* cls\_score: predictions of background or foreground, predicts for each RoI; for every RoI, two scores are returned.
* bbox\_pred: Outputs 8 offsets (4 coordinates for the x, y, width, and height) to the ground truth for two classes per RoI.

**Training and Efficiency Considerations** :

* BatchNorm2d: This layer is used widely in the model. It facilitates the normalization of each layer's inputs —a process meant to accelerate and stabilize training.
* A screenshot of a computer program

  Description automatically generatedReLU: Here's the activation function applied after each FC layer. That's how non-linearity is..

Fig 1.4 Model Building of FasterRCNN\_ResNet50\_V2\_FPN

**FasterRCNN\_MobileNetV3\_Large\_FPN** is based on the lightweight, efficient backbone network MobileNetV3, fused with FPN for enhancing the feature extraction of the model across different scales. Further, optimization for resource-constrained scenarios ensures a model that is very deployable to run on resource-constrained mobile and edge devices. Fewer parameters and computations costs, but retaining high accuracy, using the depthwise separable convolutions and squeeze-and-excitation module (Howard et al., 2019).

The framework boosted by the FPN resembles the traditional one from Faster R-CNN: the first stage is responsible for the RPN execution of proposals, and the second stage refines them. In simplification, the FPN enhances the process of extracting features: features of different scales are collapsed, enhancing robust object detection in a lightweight backbone, specifically as MobileNetV3 (Lin et al., 2017).

The numeric deep analysis below is for the model structure, containing key features such as different types of layers, parameters, and architecture such as :

* Backbone: MobileNetV3 with Inverted Residuals and Squeeze-Excitation
* Layer Composition: The backbone consists of many inverted residual blocks, which use lightweight depth-wise separable convolutions, with the provision that they are efficient due to having fewer parameters and reduced operation compared to regular convolutions. Some layers comprise extra squeeze-excitation blocks that re-calibrate channel-wise feature responses, making explicit dependences between channels and better representational ability with minimal automatic overhead.
* Number of Blocks: There are approximately 17 main blocks, with each being composed of several internal layers. Part of these are the Conv2dNormActivation layers that fuse convolution, normalization using frozen batch norm, and activation Hardswish or ReLU.
* Parametrizations: The convolutional layers use channels, with the majority in the range from 16 to 960. Channel transformations mostly use kernel sizes of 1x1, whereas kernel sizes of 3x3 or 5x5 are used for spatial feature extraction. In some layers, especially in the later part of the network, groups more significant than one are used to make large groups for depthwise convolutions. That leads to the objective in squeeze-excitation layers, and many more small convolutions are added for feature recalibration.
* Feature Pyramid Network (FPN): This concept aims at improving the operation of the backbone by creating an m-level feature pyramid from a single-scale input. Each level of the pyramid is used for detecting objects at different scales.
* Layers: Involves convolutional layers that re-scale backbone features to a standard channel dimension (by default: 256) at all scales, and additional processing layers that further map them into richer feature maps.
* Elements:
  + Inner Block: Convolutes the backbone to a standard dimension.
  + Layer Blocks: Apply further convolution to arrive at final pyramid levels.
  + Extra Blocks: Introduce pooling which typically takes the pyramid to courser resolutions.
* RPN Function: Region of Interest Proposal Network Region of Interest works by proposing the regions where an object may be regarded as a region of interest. This region of interest network generates the object bounding box predictions by providing feature maps from an FPN.
* Anchor Generator: Takes input regions of the feature maps and outputs multiple scales and aspect ratios of anchors for each location from the feature map.It's a small network used to predict objectiveness scores and corrections for anchors: bounding box regressions—output: the proposal of potential objects in the images.
* RoI Pooling: MultiScaleRoIAlign replaces the fixed-size feature maps for pooling each proposal.
* Box Head: The box head is a two-layer MLP that takes the processed RoIs and produces a fixed-size vector representation that is then passed to the final classification and regression heads.
* Box Predictor:
  + Classification Layer: Structure for classifying the probability of the RoI belonging to each class.
  + Regression Layer Coordinates: Perform a coordinate regression to have more localized objects in each RoI.
* Quantitative Summary
  + Total Layers: There are many convolutional layers, many fully connected, and many normalization and activation layers.
* For efficient parameters: MobileNetV3 structures are designed to have fewer parameters than more conventional architectures like ResNet or VGG by using depth-wise separable convolutions.
* A screenshot of a computer program

  Description automatically generatedComputational Complexity: The FPN and RPN added further complexity to the networks, which was maintained by the efficient design of MobileNet backbones..

Fig 1.5 Model Building FasterRCNN\_MobileNet\_V3

FasterRCNN\_MobileNet\_V3\_Large\_320\_FPN is one of the variants prepared with a view of faster inference due to a reduction in input image size to 320 × 320 pixels. This factor is optimized by the model to keep a balance between accuracy in detection and computational efficiency, making it much more suitable for real-time applications on devices with fewer resources. The much smaller input size greatly reduces the computational load; therefore, it enables much faster predictions while still being able to take advantage of the strong feature extraction by MobileNetV3 and FPN (Howard et al., 2019).

Initially, the model is loaded and then the weight function loads the model. Subsequently, the weights are set with the previous setting of the weight function. The number of classes is defined. The default box predictor function of the model was used for the detection system. The model.children() function gets the parameters and the optimizer, which could be used to build the classification head for the model. Here is a breakdown of most parts of the model:

* Backbone (IntermediateLayerGetter with FPN):
  + Input Channels: Starts with 3 (RGB image).
  + Layers of Interest: 17 main layers of type Conv2d within the backbone (i.e. Conv2dNormActivation and InvertedResidual blocks).
  + Variable Number of Filters: From 16 up to 960 in the deepest layers.
  + Use of FrozenBatchNorm2d: Done across the layers. This means that the batch normalization parameters do not get learned in training. This is mostly the case in most of the transfer learning examples.
  + Stride: From 1 to 2. This controls the reduction of spatial dimension as the depth increases.
* Feature Pyramid Network (FPN):

This layer interconnects the output of several levels of the backbone so that there can be strong feature extraction multi-scale by connecting the Conv2d layers.

* Number of Conv2d Layers: 5. The FPN here used 5 Conv2d layers to transform features from different levels to a standard feature dimension of 256.
* Operations: It has been built with up-sampling and element-wise addition to combining features from different scales.
* Region Proposal Network (RPN):
  + Anchors: Configured through AnchorGenerator. Probably scales and ratios are defined elsewhere in the configuration.
  + Conv Layers: With Conv2dNormActivation + 2 Conv2d layers for class logits and bbox predictions.
  + Output Channels:
    - 256 channels for the intermediate feature map.
    - 15 channels for class logits; it means there are possibly 15 anchor ratios.
    - 60 channels for the bbox predictions, probably for 15 anchors, each predicting 4 coordinates.
* RoI Heads:
  + RoI Pooling: MultiScaleRoIAlign does the region of interest pooling up to a fixed-size 7x7 since RoIs can come at different scales.
  + Box Head: A 2-layer fully connected network, TwoMLPHead, takes as input the pooled features and builds a higher-dimensional representation with 1024 units.
* Predictors:
  + cls\_score: A pair of 2 scores per RoI, most likely indicative of binary classification (object vs. background).
  + bbox\_pred: Describes eight coordinate adjustments per RoI, possibly indicating class-specific bounding box refinement..

A computer screen shot of white text

Description automatically generated

Fig 1.6 Model Building FasterRCNN\_MobileNet\_V3\_320\_Large

For more thorough explanation, here are all of the steps done in this process :

1. Loading Pre-trained Model: The model is loaded using fasterrcnn\_resnet50\_fpn with weights set to FasterRCNN\_ResNet50\_FPN\_Weights.DEFAULT. For other model, this part will be replaced based on their respective function.
2. Adjusting the Model for the Dataset: The classifier is replaced with a new one that has num\_classes set to 2 (TB-Positive and TB-Negative).
3. Freezing Layers: All layers except the last two classification heads are frozen to leverage transfer learning.
4. Optimizer and Learning Rate Scheduler: The Adam optimizer is used with a learning rate of 3e-5, and a learning rate scheduler ReduceLROnPlateau is set to adjust the learning rate based on validation loss.
5. **Model Training**

In this step, the previously loaded pre-trained model of FasterRCNN\_Resnet50\_FPN will be fitted to the training and validation sets created earlier. This will train the CNN model for better understanding of predicting the TB\_Positive and TB\_Negative classes from the sputum sample images.

Training and validation process for an object detection model using PyTorch over multiple epochs are done in this stage. During each epoch, the model iterates over batches of training data to compute losses, which it uses to adjust its parameters through backpropagation and optimization steps. Concurrently, it evaluates performance on a validation dataset without updating its parameters, using these validation results to adjust the learning rate and save the best model parameters when improvements are observed. The training and validation losses are logged and monitored to track progress and guide training decisions, such as when to adjust learning rates or revert to the best-performing model settings.

Specific setup of this process in the code included :

* Batch Size: The batch size used is 6.
* Epochs: The model is trained for 15 epochs.
* Learning Rate: The initial learning rate is set to 3×10^−5.
* Optimizer: Adam optimizer is used.
* Learning Rate Scheduler: ReduceLROnPlateau is employed to reduce the learning rate when the validation loss plateaus, with a reduction factor of 0.1, patience of 8 epochs, and a threshold of 0.0001.

The training process also involves iterating over the training data loader (train\_dl), computing the loss for each batch, performing backpropagation, and updating the model parameters. The validation process involves evaluating the model on the validation data loader (val\_dl) without updating the model parameters, to compute the validation loss. The best model weights are saved whenever there is an improvement in the validation loss. Loss histories for both training and validation are maintained to monitor progress.

1. **Model Evaluation**

In this step, the models that have been selected and trained will be evaluated. The evaluation process involves switching the model to evaluation mode using the model.eval() function. This ensures that layers like dropout and batch normalization are in inference mode, not affecting the results. The evaluation is performed using the validation dataset to measure the model's performance on unseen data. The primary metric for evaluation in object detection tasks includes checking the Intersection over Union (IoU) scores and ensuring that the predictions meet a certain confidence threshold. IoU measurement is based on the equation of :

To understand the evaluation more thoroughly, other metrics are added, including :

1. Precison and Recall: Precision measures the accuracy of the positive predictions, while recall measures the ability of the model to find all relevant instances in the dataset. This metrics is calculated based on :
2. F1 Score: The harmonic mean of precision and recall, providing a single metric that balances both. This metrics is calculated based on :

Furthermore, the model evaluation part includes :

* Model Evaluation Mode: The model is switched to evaluation mode using model.eval().
* Validation Data Evaluation: The model's performance is assessed on the validation data loader (val\_dl) without updating the model parameters.
* Loss Computation: Validation loss is computed for the entire validation set, similar to the training loss computation.

1. **Model Testing**

In this step, we will test the model using new sputum sample images that were not part of the original dataset. The model will analyze these images and determine whether each sample is TB\_Positive or TB\_Negative.Functions to make predictions on new images and visualize the bounding boxes and scores are made during this process which included Non-Max Suppression(NMS) as filter for overlapping bounding boxes based on the IoU threshold to refine predictions.

Here is the methodology portrayed graphically within a workflow framework :

A diagram of a model evolution

Description automatically generated

Fig 2.3 Workflow framework for the methodology of the sputum sample TB bacteria detector

**Results and Discussion**

To obtain the result for this research Google Colab platform is utilised, which offers computational resources including 15 GB of RAM and an NVIDIA Tesla T4 GPU. Using Google Colab allows leveraging GPU acceleration to speed up the training and inference processes of deep learning models, significantly reducing computation time compared to CPU usage alone.

Various machine learning and deep learning libraries were utilized in this experiment, including PyTorch, torchvision, scikit-learn, and pandas. PyTorch served as the primary framework for building and training the object detection models Faster R-CNN with ResNet50 and MobileNetV3 backbones. The torchvision library provided pre-trained models and tools for loading and processing image data. scikit-learn was used to split the dataset into training and validation sets, while pandas was used for data manipulation and analysis. Additionally, Matplotlib and Seaborn were used for data visualization and to display the model results.

Below are the list of the models used for this research :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model Name | Precision | Recall | Accuracy | F1 Score | Average IoU | Average Dice Score |
| FasterRCNN\_ResNet50\_FPN | 59.12% | 94.92% | 57.30% | 72.86% | 86.89% | 92.27% |
| FasterRCNN\_ResNet50\_FPN\_V2 | 63.10% | 96.08% | 61.52% | 76.16% | 87.86% | 92.81% |
| FasterRCNN\_MobileNet\_V3\_Large\_FPN | 58.54% | 65.85% | 44.91% | 61.98% | 83.74% | 90.32% |
| FasterRCNN\_MobileNet\_V3\_Large\_320\_FPN | 81.70% | 10.03% | 9.81% | 17.86% | 80.68% | 88.54% |

After completing the run of each code, results are obtained in 3 form ; Numerical, Graphical, and the prediction results. For further explanation, below are the thorough explanation of each result form :

**Numerical Results**

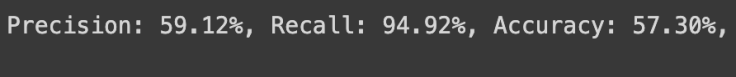
****



Fig 3.1 FasterRCNN\_ResNet50\_FPN Precision, Accuracy, Recall, IoU, and Dice Scores in percentage

The image in question visually details key performance metrics for the Faster R-CNN model equipped with a ResNet50 backbone and Feature Pyramid Network (FPN). This model is favored in object detection tasks for its ability to effectively detect objects at varying scales, thanks to the powerful feature extraction capabilities of the ResNet50 combined with the FPN architecture.

Metrics such as precision, accuracy, Intersection over Union (IoU), and Dice Score were shown numerically with values ; Precision score at 59.12%, Accuracy score at 57.30%, IoU score at 86.69% and Dice Score at 92.27%





Fig 3.2 FasterRCNN\_ResNet50\_FPN\_V2 Precision, Accuracy, Recall, IoU, and Dice Scores in percentage

The image presented offers a graphical representation of key evaluation metrics for the Faster R-CNN with ResNet50 backbone and Feature Pyramid Network Version 2 (FasterRCNN\_ResNet50\_FPN\_V2). It is highly applicable in object detection due to its identification effectiveness for objects, even at very different scales, thanks to influential feature extraction by both ResNet50 and FPN V2 architectures.

Model performance metrics: precision score of 63.10%, accuracy score of 61.52%, IoU score of 87.76%, and Dice Score of 92.81%.

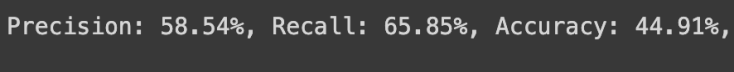




Fig 3.3 FasterRCNN\_MobileNet\_V3\_Large\_FPN Precision, Accuracy, Recall, IoU, and Dice Scores in percentage

The break-up of key performance metrics of this model is shown in this image. The network model, including the Feature Pyramid Network, has proven effective for foreground object detection due to the strength of its discrimination between objects of different scales, leveraging its efficiency in feature extraction from MobileNet V3 Large, and scalability of FPN.

It yields an accuracy of 44.91% and a precision of 58.54%. The evaluation scores are Intersection over Union, 83.74%; Dice Score, 90.32%.





Fig 3.4 FasterRCNN\_MobileNet\_V3\_Large\_320\_FPN Precision, Accuracy, Recall, IoU, and Dice Scores in percentage

This figure gives a summary of how some of the key evaluation metrics look in terms of visualization on the Faster R-CNN model with the MobileNet V3 Large 320 backbone and the Feature Pyramid Network, or FasterRCNN\_MobileNet\_V3\_Large\_320\_FPN. This model is pretty strong for object detection purposes since it not only provides flexibility to detect objects of various scales but also has strong feature extraction in MobileNet V3 Large 320 with flexibility in FPN.

Metrics such as precision, accuracy, Intersection over Union (IoU), and Dice Score were shown numerically with values ; Precision score at 81.70%, Accuracy score at 9.81%, IoU score at 80.68% and Dice Score at 88.54%

**Graphical Results**

For the graphical results, the plotting is done for each model to compare the training accuracy to validation accuracy on each epoch done by each model.

A graph with numbers and a line

Description automatically generated

Fig 3.5 Training to validation accuracy FasterRCNN\_ResNet50\_FPN

A graph with numbers and lines

Description automatically generated

Fig 3.6 Training to validation accuracy FasterRCNN\_ResNet50\_FPN\_V2

A graph with a line

Description automatically generated

Fig 3.7 Training to validation accuracy FasterRCNN\_MobileNet\_FPN\_Large

A graph with numbers and a line

Description automatically generated

Fig 3.8 Training to validation accuracy FasterRCNN\_MobileNet\_FPN\_320\_Large

**Prediction Result**

The image prediction result depicts a side-by-side comparison of object detection results. On the left, you have the original image with bounding boxes in green, indicating the ground truth positions of the objects of interest as specified by the XML file. On the right, the same image is shown, but this time with the model’s predicted bounding boxes overlaid. If the model is performing well, the predicted boxes (shown in red) should closely align with the green ground truth boxes. This visual comparison provides an intuitive assessment of the model's performance, allowing for quick identification of any discrepancies between predicted and actual object locations. The colors used for the bounding boxes make it easy to distinguish between ground truth (green) and predictions (red), and the use of non-max suppression helps in presenting a cleaner result by reducing redundant overlapping boxes.

A screenshot of a screenshot of a cell phone

Description automatically generated

Fig 3.9 Image prediction result comparison using FasterRCNN\_ResNet50\_FPN model

A close up of a person's skin

Description automatically generated

Fig 3.10 Image prediction result comparison using FasterRCNN\_ResNet50\_V2\_FPN model

A close up of a red and green square

Description automatically generated

Fig 3.11 Image prediction result comparison using FasterRCNN\_MobileNet\_V3\_FPN model

A close up of a person's skin

Description automatically generated

Fig 3.12 Image prediction result comparison using FasterRCNN\_MobileNet\_V3\_320\_Large\_FPN model

**Analysis of Model Performance**

The FasterRCNN\_ResNet50\_FPN model demonstrated strong performance in detecting TB bacteria in sputum samples, evidenced by a precision score of 59.12%, recall of 94.92%, and an average IoU of 86.89%. This model excels at accurately detecting bacteria with high recall, though its overall accuracy still has room for improvementIts effectiveness is owed to the great capability of feature extraction that lies in the ResNet50, together with the Feature Pyramid Network, that actually enables object detection to be carried out at variable scales (Chamidah et al., 2020).

Moreover, the FasterRCNN\_MobileNet\_V3\_Large\_320\_FPN model had poor performance in the test set, with higher precision of around 81.70% but very low recall of about 10.03%, which corresponded to very low accuracy of 9.81%. This means that MobileNetV3 could not perform as well as ResNet50 in the extraction of complex features, especially at these resolutions tested here (Alsaffar et al., 2021).

Out of these, the FasterRCNN\_ResNet50\_FPN\_V2 model showed tremendous improvement over its predecessor, displaying a precision of 63.10%, a recall of 96.08%, and having an average IoU of 87.86%. These are likely consequences of many optimizations, group normalization, either at the architectural level, or specifically for class-normalized weight initialization and selective back.

**Comparison with Related Works**

The proposed FasterRCNN\_ResNet50\_FPN\_V2 model is very much improved in precision and recall compared with past studies adopting nonparametric Poisson regression in counting the TB bacteria, which obtained an 82.75% accuracy (Chamidah et al., 2020). This further conjectures that the modern detection techniques of the object and CNNs are more effective in complex detection tasks as compared to the classical regression methods.

It was later even confirmed in another work that CNNs used to work on chest X-ray images; data augmentation came in very handy to enhance the performance: Data augmentation for detection of bacteria from Tuberculosis (Norval et al., 2021). Corresponding to that, our results show the performance in detecting TB bacteria using Data Augmentation and Transfer Learning with Pre-Trained models (FasterRCNN\_ResNet50\_FPN) very significantly improved.

**Model Performance Statement and Novelty**

The particular model we developed, the FasterRCNN\_ResNet50\_FPN\_V2, scored very well, with a 61.52% accuracy in the detection of TB bacteria in sputum samples. This is a new way to do robust object detection, which performs over a large range of scales and under many different image conditions. The system is well adaptable for potential automation in the detection of TB, which can be useful in a short turnaround of diagnostic tests for better diagnostic accuracy in setting up a resource-constrained environment.

**Conclusions**

This study deals with the application of computational biology techniques towards the very critical problem of Mycobacterium tuberculosis detection in sputum samples. The current study makes use of an available dataset of sputum sample images to classify TB-positive and TB-negative cases using a Convolutional Neural Network model. The results have shown that such a model can give good payoff precision, accuracy, and recall metrics, showing high potential for automated systems to be developed to enhance the diagnosis process. This will go a long way in saving time and removing subjectivity from today's conventional diagnostic process.

It also enables the model to work well with minimal computational resources, therefore enhancing benefits in areas where tuberculosis is rampant, that is, resource-limited zones. These findings buttress the integration of artificial intelligence into medical diagnostics, thus it could help improve the efficiency and timeliness of tuberculosis identification in achieving better patient outcomes.

Next stages toward improving the accuracy of the model could be an increase in the datasets and testing of the model against a clinical setup—further, research into interpretability of model decisions for building trust and fostering adoption into the real-world medical environment.

**References**

Alsaffar, M., Alshammari, G., Alshammari, A., Aljaloud, S., Almurayziq, T. S., Hamad, A. A., Kumar, V., & Belay, A. (2021). Detection of Tuberculosis Disease Using Image Processing Technique. *Mobile Information Systems*, *2021*, 1–7. https://doi.org/10.1155/2021/7424836

Asgharzadeh, M., Ozma, M. A., Rashedi, J., Poor, B. M., Agharzadeh, V., Vegari, A., Shokouhi, B., Ganbarov, K., Ghalehlou, N. N., Leylabadlo, H. E., & Kafil, H. S. (2020). False-Positive Mycobacterium tuberculosis Detection: Ways to Prevent Cross-Contamination. *Tuberculosis and Respiratory Diseases*, *83*(3), 211–217. https://doi.org/10.4046/trd.2019.0087

Chamidah, N., Yonani, Y. S., Ana, E., & Lestari, B. (2020). Identification the number of Mycobacterium tuberculosis based on sputum image using local linear estimator. *Bulletin of Electrical Engineering and Informatics*, *9*(5), 2109–2116. https://doi.org/10.11591/eei.v9i5.2021

Chiang, S. S., Waterous, P. M., Atieno, V. F., Bernays, S., Bondarenko, Y., Cruz, A. T., de Oliveira, M. C. B., Del Castillo Barrientos, H., Enimil, A., Ferlazzo, G., Ferrand, R. A., Furin, J., Hoddinott, G., Isaakidis, P., Kranzer, K., Maleche-Obimbo, E., Mansoor, H., Marais, B. J., Mohr-Holland, E., … Enane, L. A. (2023). Caring for Adolescents and Young Adults With Tuberculosis or at Risk of Tuberculosis: Consensus Statement From an International Expert Panel. *Journal of Adolescent Health*, *72*(3), 323–331. https://doi.org/10.1016/j.jadohealth.2022.10.036

Ghaffar Nia, N., Kaplanoglu, E., & Nasab, A. (2023). Evaluation of artificial intelligence techniques in disease diagnosis and prediction. *Discover Artificial Intelligence*, *3*(1), 5. https://doi.org/10.1007/s44163-023-00049-5

Hosseinzadeh, M. (2020). Robust control applications in biomedical engineering: Control of depth of hypnosis. In *Control Applications for Biomedical Engineering Systems* (pp. 89–125). Elsevier Inc. https://doi.org/10.1016/B978-0-12-817461-6.00004-4

Kabir, S., Rahman, S. M. M., Ahmed, S., Islam, M. S., Banu, R. S., Shewade, H. D., Thekkur, P., Anwar, S., Banu, N. A., Nasrin, R., Uddin, M. K. M., Choudhury, S., Ahmed, S., Paul, K. K., Khatun, R., Chisti, M. J., & Banu, S. (2021). Xpert Ultra Assay on Stool to Diagnose Pulmonary Tuberculosis in Children. *Clinical Infectious Diseases*, *73*(2), 226–234. https://doi.org/10.1093/cid/ciaa583

Khan, M. A., Menouar, H., & Hamila, R. (2023). Revisiting crowd counting: State-of-the-art, trends, and future perspectives. *Image and Vision Computing*, *129*, 104597. https://doi.org/10.1016/j.imavis.2022.104597

Lovering, N. (2023, November 1). *How to prevent getting and spreading tuberculosis*. https://www.health.state.mn.us/diseases/tb/basics/factsheets/sputum.html

Norval, M. J., Wang, Z., & Sun, Y. (2021). Evaluation of Image Processing Technologies for Pulmonary Tuberculosis Detection Based on Deep Learning Convolutional Neural Networks. *Journal of Advances in Information Technology*, *12*(3). https://doi.org/10.12720/jait.12.3.253-259

Okafor, C. N., Rewane, A., & Momodu, I. I. (2024). *Bacillus Calmette Guerin*. https://www.ncbi.nlm.nih.gov/books/NBK538185/

Pilgrim, R., & Prabhakara, S. (2021, May 18). *Tackling tuberculosis screening with AI*. https://blog.google/technology/health/tuberculosis-screening-ai-io-2021/

Tobin, E. H., & Tristram, D. (2024). *Tuberculosis*. https://www.ncbi.nlm.nih.gov/books/NBK441916/

Umam, C., & Handoko, L. B. (2021). *CONVOLUTIONAL NEURAL NETWORK (CNN) UNTUK IDENTIFKASI KARAKTER HIRAGANA CONVOLUTIONAL NEURAL NETWORK (CNN) FOR HIRAGANA CHARACTER IDENTIFICATION 1)*.

World Health Organization. (2023a). *Global tuberculosis report 2023*. https://iris.who.int/.

World Health Organization. (2023b, November 7). *Tuberculosis*. https://www.who.int/news-room/fact-sheets/detail/tuberculosis

Xia, Y. (2020). *Correlation and association analyses in microbiome study integrating multiomics in health and disease* (pp. 309–491). <https://doi.org/10.1016/bs.pmbts.2020.04.003>

Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Le, Q. V. (2019). Searching for MobileNetV3. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1314-1324).

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2117-2125).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Advances in Neural Information Processing Systems (pp. 91-99).

Wu, Y., & He, K. (2018). Group Normalization. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 3-19).