Malaria Cell Report

1.Introduction: - Sathya

1.1. Motivation

1.3 Application

2.Literature Review - Niel

3. Gap Analysis - Niel

4. Proposed Methodology -Sai Charan

5. Results - all

References:

Introduction

In the past few years, a single virus has halted the world’s progression and truly cast a light on how underequipped we are in fighting the war against illnesses. While the COVID virus will soon be prevented by a vaccine, there are many diseases that will continue to plague the world for decades to come. The most pressing of these diseases would have to be Malaria. Malaria is brought about by the infection of the Plasmodium parasite. The female Anopheles mosquito acts as a vector and transmits the parasite into a human. An infected human can in turn infect another Anopheles mosquito which would just feed into the cycle. With no sure-fire way of shielding us from mosquito bites, the infection is one that can barely be prevented. The infection would then bring upon the onset of fever, chills, headaches and in more serious cases, yellow skin, seizures, and coma. If left completely untreated, it can even progress to death within just a period of 24 hours. Luckily, humans have made marvelous strides in treating this illness. The diagnosis includes a blood test to scan for the presence and the type of malaria present. Blood smears are the most common and accurate test for the diagnosis of malaria. The attending physician will take a sample of the blood and the sample is sent to a lab where it is stained and observed (manually / by a human) under a microscope. The treatment depends on the individual but it usually consists of a prescription of an antimalarial drug.

## Motivation

The true severity of malaria is often overlooked due to its lack of impact in developed countries. Malaria is still as deadly as ever, with Africa shouldering the highest mortality rate to the disease. From its infection of patient 0 back in 2019, COVID has infected around a total of 328 million people. On the other hand, malaria had infected 241 million people in 2020 alone, and a devastating 95% of these cases were recorded only in Africa. Furthermore, 80% of the 627,000 deaths were of children below the age of 5. These horrifying numbers would only continue to grow and hence require much-needed intervention. While the treatment is out of the scope of computer science, it surely can optimize the time taken to diagnose the disease while simultaneously increasing the accuracy of said diagnosis. While the pre-existing method for diagnosis has proven to be moderately effective, it can be seen that there is a lot of room for error. There is always the risk that the test will return a false positive or a false negative, both of which engender their own negative consequences. There is also the added risk that the diagnosis would be too late to treat the disease effectively. Automating this process will have greater precision in less time.

## Application

We intend to design a model using CNN that would take the blood smear images as input and identify the presence and the type of the plasmodium parasite. The detection would yield extremely high accuracy as there is little to no margin for error. The deployment of our model would be primarily focused on the regions of Sub-Saharan Africa, Latin America, and Asia. These regions are home to a lot of under-developed / developing countries. Our highest priority hence is to make sure that our model is lightweight enough so that it can be run on affordable hardware. We intend to achieve this by using Depthwise separable convolutional layers which is further discussed in the proposed methodology section. We intend to take any load off of the technicians or physicians that have been working endlessly with COVID running rampant in the past few years. Using lightweight CNNs to detect malaria would increase the rate of diagnosis and help curb the disease. The current smear tests have an accuracy of 85% detecting infected cells. We hope to achieve a larger number using our model.

# Literature Review

Applying faster R-CNN for object detection on malaria images (Hung, J., & Carpenter, A. [1]) for the first time applies an object detection model previously used on natural images to identify cells and recognize their stages in brightfield microscopy images of malaria-infected blood. Many micro-organisms like malaria parasites are still studied by expert manual inspection and hand counting. This type of object detection task is challenging due to factors like variations in cell shape, density, and color, and uncertainty of some cell classes. In addition, annotated data useful for training is scarce, and the class distribution is inherently highly imbalanced due to the dominance of uninfected red blood cells. Faster Region-based Convolutional Neural Network (Faster R-CNN) is used, one of the top performing object detection models in recent years, pre-trained on ImageNet but fine tuned with the data, and compare it to a baseline, which is based on a traditional approach consisting of cell segmentation, extraction of several single-cell features, and classification using random forests.The Faster R-CNN outperforms the baseline and put the results in context of human performance.

Clustering-based dual deep learning architecture for detecting red blood cells in malaria diagnostic smears (Kassim, Y. M., Palaniappan, K., Yang, F., Poostchi, M., Palaniappan, N., Maude, R. J., ... & Jaeger, S. [2]) The proposal is a novel pipeline for red blood cell detection and counting in thin blood smear microscopy images, named RBCNet, using a dual deep learning architecture. RBCNet consists of a U-Net first stage for cell-cluster or superpixel segmentation, followed by a second refinement stage Faster R-CNN for detecting small cell objects within the connected component clusters. RBCNet uses cell clustering instead of region proposals, which is robust to cell fragmentation, is highly scalable for detecting small objects or fine scale morphological structures in very large images, can be trained using non-overlapping tiles, and during inference is adaptive to the scale of cell-clusters with a low memory footprint. It was tested on an archived collection of human malaria smears with nearly 200,000 labeled cells across 965 images from 193 patients. Cell detection accuracy using RBCNet was higher than 97\%. The novel dual cascade RBCNet architecture provides more accurate cell detections because the foreground cell-cluster masks from U-Net adaptively guide the detection stage, resulting in a notably higher true positive and lower false alarm rates, compared to traditional and other deep learning methods. The RBCNet pipeline implements a crucial step towards automated malaria diagnosis.

A Malaria Diagnostic Tool Based on Computer Vision Screening and Visualization of Plasmodium falciparum Candidate Areas in Digitized Blood Smears ( Linder, N., Turkki, R., Walliander, M., Mårtensson, A., Diwan, V., Rahtu, E., ... & Lundin, J. [3]) manual evaluation of blood films is highly dependent on skilled personnel in a time-consuming, error-prone and repetitive process. In this study the proposed method is using computer vision detection and visualization of only the diagnostically most relevant sample regions in digitized blood smears.Giemsa-stained thin blood films with P. falciparum ring-stage trophozoites (n = 27) and uninfected controls (n = 20) were digitally scanned with an oil immersion objective (0.1 µm/pixel) to capture approximately 50,000 erythrocytes per sample. Parasite candidate regions were identified based on color and object size, followed by extraction of image features (local binary patterns, local contrast and Scale-invariant feature transform descriptors) used as input to a support vector machine classifier. The classifier was trained on digital slides from ten patients and validated on six samples.

A semi-automatic method for quantification and classification of erythrocytes infected with malaria parasites in microscopic images (Gloria Díaz, Fabio A González, Eduardo Romero [4]). In this study, a three phase approach is employed. The first phase involves image processing. The images are corrected for luminance differences produced by the acquisition process by means of a local adaptive low-pass filter. The second phase is that of the erythrocyte recognition phase. This phase is further split up into three steps. In the first step, a color pixel classification process uses a machine-learning strategy for classifying a particular color space, which is then used as a look-up table for labeling each pixel as either foreground or background. Secondly, pixels labeled as foreground are grouped into one simplified Inclusion-Tree. The resulting tree is simplified to satisfy the restrictions imposed by the erythrocyte morphological structure and their spatial relationships. Finally, clumped shapes, a possible result of the binarization process, are split using an efficient template matching strategy. This approach searches for the better matching between a chain code representation of the clumped shape contour and an ideal erythrocyte, estimated from the original image by an Expectation-Maximization algorithm.The last phase is the classification of erythrocytes among any of the four possible classes. This classification is achieved upon 25 features which correspond to the statistics of the distributions of color, texture, illumination and edges, extracted from the erythrocytes detected in the previous step. The whole process consists of two steps: firstly, a binary classifier decides whether the erythrocyte is healthy or not. Then a multiclass classifier strategy assigns each erythrocyte to one of the three infection life stages: ring stage, trophozoite or schizont.

Parasite detection and identification for automated thin blood film malaria diagnosis ( F. BorayTeka, Andrew G.Dempster, İzzet Kale [5]). The proposal is that of a novel binary parasite detection scheme that is based on a modified K nearest neighbour (KNN) classifier which provides an adjustable sensitivity–specificity parasite detection. This study affirms the applicability of the method to malaria diagnosis by comparing its results to an expert microscopist’s ideal detection performance from a Bayesian perspective. Three different classification schemes for identification are compared. The conclusion states that detection, and species and lifecycle-stage identification tasks can be performed successfully by a single multi-class classification. It also states that the necessity of seeking a high-level generalization in two assumed categories can be removed. The study implements a total of 8 stages for the detection and classification: Stages 1-5: Preprocessing; Stage 6: Object Extraction; Stage 7: Feature Extraction; Stage 8 : Classification using K nearest neighbor clustering algorithm.

Malaria Cell Detection Using Evolutionary Convolutional Deep Networks (Qin, B., Wu, Y., Wang, Z., & Zheng, H. [6]) uses evolutionary convolutional deep networks. It can work with keras, it automatically generates a good neural architecture. It can be called as sub domain of AutoML. The data used used here is by NIH. Around 28000 images of both infected and uninfected cells. Pre processing of dataset included sample purification, image rescaling and data enhancement. The final accuracy of the model is 99.98 percent. Further work includes testing accuracy of model on different datasets, improving mobility and testing with better network architectures.

CNN-Based Image Analysis for Malaria Diagnosis (Liang, Zhaohui and Powell, Andrew and Ersoy, Ilker and Poostchi, Mahdieh and Silamut, Kamolrat and Palaniappan, Kannappan and Guo, Peng and Hossain, Md Amir and Sameer, Antani and Maude, Richard James and Huang, Jimmy Xiangji and Jaeger, Stefan and Thoma, George [7]) looks into normal CNN models and transfer learning models. The CNN model is 17 layers and gives an average accuracy of 97.37\%. The transfer learning model gives an accuracy of 91.99\% on the same dataset. The dataset used here has been acquired from Chittagong Medical College Hospital, Bangladesh. It contains 27578 images of infected and uninfected cell. The dataset is perfectly balanced. The training-testing split used here is 90\%-10\%. The results indicate that the new CNN model is more accurate than the transfer learning model (by around 7\%). The limitation is that accuracy is relatively lower than models in other papers.

In Image Classification of Unlabeled Malaria Parasites in Red Blood Cells (Zhang, Z., Ong, L. S., Fang, K., Matthew, A., Dauwels, J., Dao, M., & Asada, H. [8]), HOGs features extracted and classifier trained offline. Viola-Jones object detection is implemented. Model out-performs PCA feature classification by 50\% and Hugh transform algorithms by 24\%.. Accuracy achieved with model is 93\%. Limitations are that red blood cells which were shriveled up were not detected. Also, the processing time is very high at 3.3 seconds with a margin of error of 0.4 seconds.

FPGA Implementation of CNN Algorithm for Detecting Malaria Diseased Blood Cells (Sağlam, S., Tat, F., & Bayar, S. [9]) implements FGPA (Field Programmable Gate Array) for CNN. The average computation time is 174 microseconds. Dataset was taken from US National Library of Medicine site. Testing contained 200 images. 90 diseases, 90 healthy and 20 invalid. Experimental accuracy was 94.76\% (189 of the 200 were correctly classified). Uses VHDL language for high efficiency (low energy usage and low use of embedded circuit platform hardware). Limitations are that the dataset used is very small having only 200 images. Also, accuracy is quite low when compared to other models at 94.76\%.

Automatic White Blood Cell Detection and Identification Using Convolutional Neural Network (Novoselnik, F., Grbić, R., Galić, I., & Dorić, F. [10]) uses image segmentation for detection of white blood cells. These cells are then classified using a CNN into 5 different classes. Dataset used for this was attained by medical staff of Faculty of medicine at Clinical Medical Center Osijek, Croatia. The dataset contains 412 images. Resolution is 2560x1920. Dataset is extremely unbalanced. The classifications are ‘Eosinophils’, ‘Lymphocytes’, ‘Monocytes’, ‘Neutrophils’ and ‘Unknown’. Modified LeNet-5CNN was used. It is a 7 layer CNN. ReLU is used as the activation function. Accuracy attained was 81.11\%. Limitation is that accuracy is low and the dataset used is highly imbalanced.

There have been several attempts at solving this problem of detecting malaria cells from a given cell sample. There are 2 main approaches. The first one being segmenting the image to extract the cells first and the second would be to run the CNN directly on the image.

However the first step to both these approaches is heavy pre-processing. As the images are of cell slides we need to get a clear image of the cells preset. To achieve this a variety of techniques have been employed. A few of them being passing a low pass filter to adjust difference in luminescence and even using a pixel classifier to determine whether a pixel belongs to the foreground or background.

The first method we found to be used in previous papers was to first run segmentation algorithms on the images. Various methods were deployed for this, a few of them being binary thresholding and specialized mathematical functions. A unique method we found was using an open source module called CellProfiler which identified the cell boundaries of various types of cells from erythrocytes to white and blood cells. These identified cells are then run through a CNN or a KNN algorithm to identify any malaria cells.

The second method is to run a classification algorithm directly on the image. The CNN’s had several layers with most of them having RELU and SIGMOID as activation functions. Another method being used was the Faster R-CNN architecture. Overall this method lowered the accuracy as there was a higher presence of false positives.

All the work that we have reviewed is focused on developed countries that have access to premium hardware. These regions are already equipped to combat malaria to their fullest capabilities. As mentioned above, Africa was home to 94\% of the total world's cases. Taking in other factors (such as technological infrastructure and Healthcare facilities), it is of paramount importance to ensure that the model to be run is computationally efficient. Our work focuses mainly on designing a computationally efficient model which can run on older hardware as these regions are impoverished and don't have access to the most cutting edge hardware. This would allow hospitals in more impoverished regions, that do not have cutting edge hardware, to also implement our software and help save lives.

# Proposed Methodology

We plan to achieve our goal using a 2 stepped process. The first step being to run and test a few pre-trained models like VGG-19 [16], ResNet50 [17] and InceptionV3[18].The accuracy F1 score would be noted down to be used later in our system. An-other major parameter that would be noted down would be the time to run that par-ticular model. This is essential as our aim to create a lightweight model would only be achieved if the run time is lesser compared to the already tested pre-trained models.

VGG-19 is a pre-trained network that is 19 layers deep. It normalizes and reduces di-mensions - to keep scale centralized - in terms of when we will perform Convolution Later on. The reason this is important, it is to bring everything “down in line” in a nor-malized, streamlined and orderly fashion - so we have some sense of normality. As in we want the general structure of what we are parsing to be normalized and central-ized so that we have a pre-defined boundary that we are being relative towards. The majority of the layers are MaxPooling ones or use ReLu as the activation function. ResNet50 is another pre-trained network that has 48 convolution layers 1 MaxPoollayer and 1 Average Pool layer giving us a total of 50 layers. This plain network was inspired by VGG neural networks, with the convolutional networks having 3×3 filters.However, compared to VGG nets, ResNets have fewer filters and lower complexity. InceptionV3 that started out from Googlenet intends to allow deeper neural networks without increasing the number of parameters. It replaces bigger convolutions with smaller convolutions which definitely leads to faster training. Say a 5 × 5 filter has25 parameters; two 3 × 3 filters replacing a 5 × 5 convolution has only 18 (3\*3 + 3\*3)

parameters instead. In addition smaller auxiliary CNN’s are added between layers during training.

Depthwise separable convolutional layers

In our CNN, we use depthwise separable convolutional layers. We use depthwise separable convolutional layers instead of normal convolutional layers as they use fewer computations than traditional convolutional layers.

Depthwise separable convolution splits a kernel into 2 separate kernels that do two convolutions.

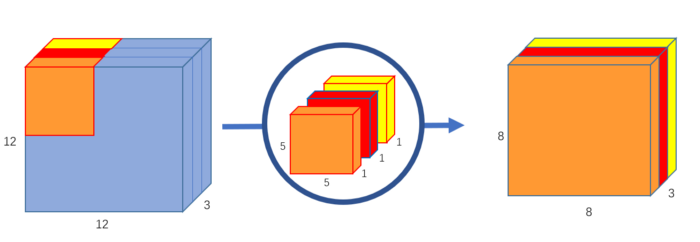
One is depthwise convolution

Another is pointwise convolution

This tends to take into account each channel, giving a higher accuracy while also being efficient.

For example, consider a 12x12x3 image.

Depthwise Convolution

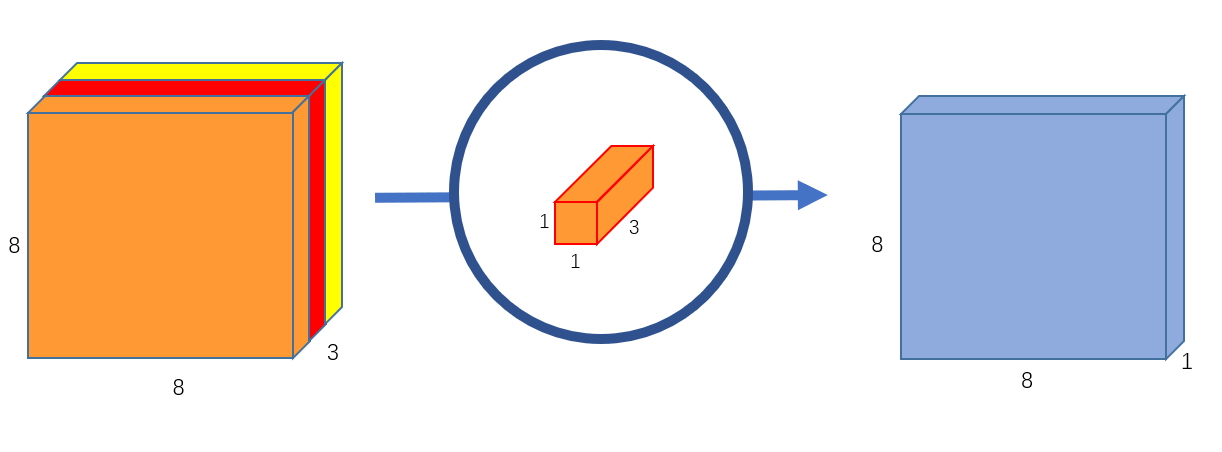


Kernel is 5x5x1. Image is 12x12x3.

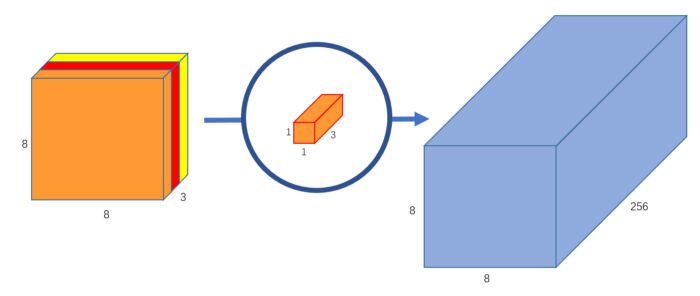
Each kernel iterates 1 channel of the image. Hence we get 8x8x1 image from each kernel.

Stacking the images from each kernel, we get a 8x8x3 image.

Pointwise Convolution



Kernel is 1x1x3. Outputs 8x8x1 image.



256 of these kernels are used.

Stacking them all on top of each other results in a 8x8x256 image

Comparison with normal convolutional layers

In a normal convolution, the 5x5x3 kernels move 8x8 times, and there are 256 of them.

That is, 5x5x3x8x8x256 = 1,228,800 multiplications.

In depthwise convolution, the 5x5x1 kernels move 8x8 times, and there are 3 of them.

That is, 5x5x1x8x8x3= 4,800 multiplications.

In pointwise convolution, the 1x1x3 kernels move 8x8 times, and there are 256 of them.

That is, 1x1x3x8x8x256= 49,152 multiplications.

Adding them, we get a total of 53,952 multiplications.

So in the end, with the normal convolutional layers, we get 1,228,800 multiplications and with depthwise separable layers we get 53,952. As we can see, fewer computations are required for the depthwise separable layers by a significant amount. Hence, the network can process more in a shorter time

Results:

Pretrained models

Accuracy, precision, recall , f1 score

Processinf time

Similarly for DS\_CNN

| Model Name | Accuracy | Precision | Recall | F1 score |
| --- | --- | --- | --- | --- |
| ResNet50 | 0.85 | 0.85 | 0.85 | 0.85 |
| VGG19 | 0.93 | 0.93 | 0.93 | 0.93 |
| InceptionV3 | 0.91 | 0.91 | 0.91 | 0.91 |
| Depthwise Separable CNN | 0.87 | 0.87 | 0.87 | 0.87 |

Time Taken

| Model Name | Time Taken |
| --- | --- |
| ResNet50 | 02:52.076058 |
| VGG19 | 06:12.583593 |
| InceptionV3 | 05:30.600212 |
| Depthwise Separable CNN | 00:38.531149 |

### Default Parameters

#### **Resnet50**

Layer (type) Output Shape Param #

=================================================================

resnet50 (Functional) (None, 2, 2, 2048) 23587712

flatten (Flatten) (None, 8192) 0

dense (Dense) (None, 512) 4194816

dropout (Dropout) (None, 512) 0

dense\_1 (Dense) (None, 1) 513

=================================================================

Total params: 27,783,041

Trainable params: 27,729,921

Non-trainable params: 53,120

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### VGG-19

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

vgg19 (Functional) (None, 2, 2, 512) 20024384

flatten\_1 (Flatten) (None, 2048) 0

dense\_2 (Dense) (None, 512) 1049088

dropout\_1 (Dropout) (None, 512) 0

dense\_3 (Dense) (None, 1) 513

=================================================================

Total params: 21,073,985

Trainable params: 21,073,985

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### 

#### Inception v3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

vgg19 (Functional) (None, 2, 2, 512) 20024384

flatten\_2 (Flatten) (None, 2048) 0

dense\_4 (Dense) (None, 512) 1049088

dropout\_2 (Dropout) (None, 512) 0

dense\_5 (Dense) (None, 1) 513

=================================================================

Total params: 21,073,985

Trainable params: 21,073,985

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### 

#### Depthwise Separable Layers model

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

==================================================================

conv2d (Conv2D) (None, 62, 62, 16) 448

max\_pooling2d (MaxPooling2D) (None, 31, 31, 16) 0

dropout\_3 (Dropout) (None, 31, 31, 16) 0

conv2d\_1 (Conv2D) (None, 29, 29, 32) 4640

max\_pooling2d\_1 (MaxPooling 2D) (None, 14, 14, 32) 0

dropout\_4 (Dropout) (None, 14, 14, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

==================================================================

separable\_conv2d (SeparableConv2D) (None, 12, 12, 64) 2400

max\_pooling2d\_2 (MaxPooling 2D) (None, 6, 6, 64) 0

dropout\_5 (Dropout) (None, 6, 6, 64) 0

separable\_conv2d\_1 (SeparableConv2D) (None, 4, 4, 64) 4736

max\_pooling2d\_3 (MaxPooling 2D) (None, 2, 2, 64) 0

dropout\_6 (Dropout) (None, 2, 2, 64) 0

flatten\_3 (Flatten) (None, 256) 0

dense\_6 (Dense) (None, 64) 16448

dropout\_7 (Dropout) (None, 64) 0

dense\_7 (Dense) (None, 1) 65

=================================================================

Total params: 28,737

Trainable params: 28,737

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_