NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

On the performance of lightweight convolutional neural networks for malaria detection

Inês Raquel Rato Borracho

Dissertation presented as partial requirement for obtaining the Master's degree in Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

ON THE PERFORMANCE OF LIGHTWEIGHT MODELS FOR MALARIA DETECTION

by

Inês Raquel Rato Borracho

Dissertation presented as partial requirement for obtaining the Master's degree in Advanced Analytics

Advisor / Co Advisor: Mauro Castelli

October 2022

DEDICATION

To my wonderful and loving family for their neverending support and encouragement.

ABSTRACT

Malaria is still a threat to millions of people. Despite the extreme importance of an early diagnosis for proper treatment in places where the disease is endemic, there is a significant disparity in terms of access to healthcare. The most widely used technique to identify malaria parasites is microscopy with Giemsa-stained blood slides. Notwithstanding its effectiveness, there are challenges in bringing it to remote places where electricity is needed, and there is a lack of skilled personnel to read the results accurately. This process can be accelerated via deep learning, as shown by extensive literature about the topic. However, many of these works are focused on performance alone, while the models are large and cannot be deployed to real-world applications. This work shows that pre-trained lightweight models such as MobileNet, MobileNetV2, NASNetMobile, and EfficientNetB0, which all were created to perform on smaller devices, can still maintain an outstanding performance despite their smaller size and having fewer parameters. Furthermore, as many weights in a network have been proven to not contribute to the result, pruning is applied on MobileNet. It is shown that the initial accuracy of 99.5% is kept as the size drastically decreases from the initial 18 MB to 5MB.

KEYWORDS

Convolution Neural Networks; Transfer Learning; Pruning; Malaria; Deep Learning; Compression

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

CNN	Convolutional Neural Network
WHO	World Health Organization
SVM	Support Vector Machine
kNN	K-Nearest Neighbors
PCR	Polymerase Chain Reaction
RDT	Rapid Diagnostic Tests
RNN	Recurrent Neural Network
NN	Neural Networks
SOM	Self-Organizing Networks
DTGCN	Deep Transfer Graph Convolutional Network
DBN	Deep Belief Network

1. INTRODUCTION

1.1. BACKGROUND CONTEXT

Malaria is a severe and often fatal disease endemic to Asia and Latin America but primarily to sub-Saharan Africa. The latter accounts for 90% of fatalities. Most of these deaths are children who age is below five years old. Other vulnerable people include pregnant women and immune-compromised patients. It is a disease that not only leads to preventable deaths and disabilities but also takes its toll on the entire family's income, contributing to the cycle of poverty. According to the World Health Organization (WHO), the direct and indirect costs of malaria in sub-Saharan Africa are over 12 billion USD per year. Malaria used to be seen as a consequence of poverty; however, there was a paradigm shift in this view as it is now interpreted as one of its causes. Impoverished and rural people are more at risk due to the lack of healthcare, diagnosis, and treatment measures. There is a collective effort to eradicate malaria, having some success in diminishing cases in some places. However, it stagnated in 2020 due to the pandemic's restrictions and difficulties. In the same years, it's estimated to have been 241 million cases worldwide; out of those, 627000 ended up in deaths, an increase of 69000 when compared to 2019. Malaria can also occur in non-endemic countries in cases of tourism of military that dislocates to tropical and sub-tropical places where the mosquito is prevalent. Since the disease is unexpected in developed countries, diagnosis and administration of anti-malarial drugs can be delayed.

Malaria is transmitted through the bite of the female *Anopheles* mosquito, a Plasmodium parasite carrier. Four parasites can infect humans: *P. falciparum, P. malariae, P. vivax, P. ovale, and P. knowlesi*. Out of these, the first one is responsible for the most severe cases. P. *vivax* and P. *ovale* can cause a less severe form of malaria but also can remain dormant for months and sometimes even years, causing a relapse later.

The first symptoms may, at first, resemble a mild virus, causing the patient not to seek immediate medical assistance. Symptoms include, but are not limited to, nausea, fever, tiredness, vomiting, muscle pain, and headaches. These symptoms usually appear seven days to 14 days after the parasite transmission. As the disease progresses, jaundice and anemia can occur. In cases of complications, cerebral malaria causes the brain to swell, which might induce coma and even death. Other complications include liver failure, shock, rupture of the spleen, low blood sugar, and acute respiratory distress syndrome. As its vector is solely the mosquito, it's worth noticing that it's not a contagious disease. Other forms of transition include from mother to fetus, blood transfusion, needle sharing, or direct contact with blood from an infected person.

The disparity in access to healthcare, in conjunction with the global efforts to eradicate malaria, implies bringing methods of diagnosis and treatment to remote places with a lack of electricity. The primary method of malaria diagnosis is a microscopy exam which requires electric power to work. There have been efforts to create lighter microscopes in conjunction with mobile phones. This is a way to tackle the lack of electrical power issue, as mobile phones can last many hours without needing to be recharged. Mobile phones are now cheap enough and have reached a level where they can perform tasks that previously solely a computer could. This implies that they can now run CAD systems. At the same time, convolutional neural networks have surpassed expectations in classification, segmentation, and detection tasks in the medical field. These algorithms have been successfully tested in apps

capable of detecting the malaria parasite in erythrocytes. There is plenty of research on the capability and efficiency of CNNs in these tasks. However, many only consider metrics like accuracy, setting the model's size aside.

Consequently, many models cannot be deployed into devices with less computer power. For instance, the state-of-the-art and popular VGG16 is a whopping 549 MB, making it inviable to use in many devices without enough memory and computer power to deploy it. There is much to explore regarding lighter-weight models designed to perform in such devices. CNNs can be created to be lightweight either through architecture, such as using depth-wise convolution or by compressing methods. These methods include knowledge distillation, quantization, huffing encoding, and pruning.

There has been some work in this field regarding the task of detecting malaria. Most of it is focused on creating models from scratch and transfer learning with models that overlook the problem of being deployable. Compression methods are, so far, less popular, leaving room for testing their performance while we compress them at different rates.

1.2. PROBLEM STATEMENT AND THESIS AIMS

In this thesis, several lightweight models will be trained to classify blood cells as either uninfected or infected with the falciparum parasite. There has been comprehensive research on the topic. However, many focus on improving accuracy and other chosen and more adequate metrics for the problem. Many of these models reach a size where deployment is no longer feasible. Other studies have focused on creating lighter models to deploy in technology with less computer power, such as smartphones. This adds the possibility of combining the diagnosis method of microscopy with an attached cell phone that will automate the process of detecting the malaria parasite in the blood cells, contributing to the elimination of difficulties posed by microscopy as a method of diagnosis in resource-constrained places. This work is dedicated to the lighter pretrained models and their performance and capability of standing to the results of previous studies. The chosen models are MobileNet, MobileNetV2, NASNetMobile, and EfficientNetB0 for their low number of parameters and smaller size than other pretrained models. The architecture of these models was conceived to perform in devices with less computer power, thus making them compatible with the goal of this thesis. Furthermore, compression methods such as pruning can diminish size by removing weights that do not contribute to the network. This method has yet to be tested in the context of the detection of malaria. So, to sum up, these are the questions and premises this work intends to answer:

- Evaluate the performance of lightweight pre-trained models on the task of malaria detection.
- Can we prune a model so that the accuracy, recall, precision, and F1- Score performance does not diminish and the model's size is reduced?

1.3. THESIS STRUCTURE

As seen above, this first chapter is dedicated to contextualizing the problem of malaria and what it represents as an often fatal disease and, as a side effect, an economic burden. Furthermore, microscopy is highlighted as a means of a practical method of diagnosis that, while being the golden standard, has room for improvement when it comes to being a suitable, portable, and automatic

method ready to be used in places with limited resources. For this achievement, convolutional neural networks need to be small to be deployed.

Chapter 2 Introduces the fundamental concepts central to the topic of neural networks, the concept of convolutional neural networks with their building blocks, and machine learning in a more general way. It will further review the concept of transfer learning, which is crucial to this thesis, and its importance when working with small datasets.

Chapter 3 will discuss microscopy as a means of diagnosis, its advantages and disadvantages, and how it can be combined with a cheap electronic device to work in conjunction to automate malaria diagnosis. Some current and past research will be reviewed with what has been done so far using machine learning techniques, with a particular focus on CNNs.

Chapter 4 offers insight into the dataset used in this work, data augmentation, and transfer learning methods. The pruning methods, hyperparameters, prevention of overfitting, and metrics used are also described. In short, this is the core description of the algorithm created.

Chapter 5 discusses the results obtained using the methods described in the earliest chapter.

Chapter 6 summarizes the work done, highlighting, and pinpointing significant conclusions.

Chapter 7 indicates the drawbacks of this work and improvements that could be added to understand what impacts the models for their success and how the dataset should be expanded to mimic the encountered real-world data. These factors are of great importance from a theoretical perspective and for deploying the models in an application capable of taking full advantage of the microscopy capabilities.

2. THEORETICAL BACKGROUND

This chapter presents some fundamental concepts of machine learning, neural networks, and deep learning. While it is not a comprehensive review, it contains the topics pertinent to this thesis.

2.1. FROM MACHINE LEARNING TO DEEP LEARNING

A subset of Artificial Intelligence that has reached tremendous success in many different industries. Machine learning is built upon a new paradigm in computer science. While hard coding was mandatory to achieve the desired result from a program, with machine learning, no direct coding was needed to achieve an output. The algorithm learns through the data that receives and discovers patterns and relationships among it, finding solutions to what would sometimes be almost impossible to code explicitly.

Data is the main word and concept to understand here. To achieve a successful result, learning demand large quantities of data. How a machine deals with the data depends on several factors, such as if it's labeled or unlabeled, quantity and quality. Based on this and the problem, several algorithms can be employed, as illustrated in figure 1.

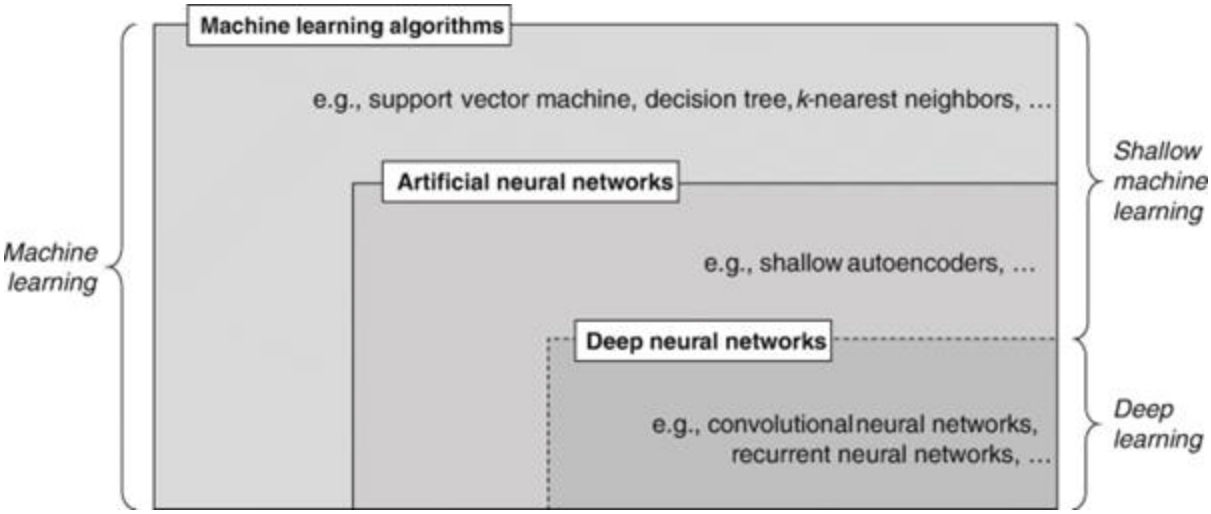

Figure 1- Several types of algorithms are widely used in machine learning. Source: Janiesch et al, 2021

Machine learning, in conjunction with the data that it is given to work with, can be divided into the following ways:

- **Supervised Learning**: This type of learning is only possible when every sample in a given dataset is assigned to a label. For example, an image of a cat must be labeled as such. A picture of a blood cell is labeled as infected or healthy for this work. As the training process advances, the model can allocate each input to its output. It can further be divided into what is known as a regression problem when the data is numerical or categorical if it is the case of a classification problem.

- **Unsupervised Learning**: Unlike the previous type, no dataset member will be assigned to a label. The algorithm will learn to associate data through means of common elements. Famous applications include customers' segmentation into specific groups, recommendation systems, and anomaly detection.

- **Reinforcement learning**: A different concept of learning. Unlike the previous, the algorithm's core contains a current state of the system, a goal, and an agent. The agent must therefore explore the environment with a series of possible actions that will be rewarded if going on the right path to achieve a goal or punished. It is a trial-and-error process. It has been successfully employed in gaming and electronic markets.

As we progress on this succinct presentation of machine learning, neural networks are approached. The main difference between more "traditional" machine learning algorithms and neural networks is feature extraction. Machine learning requires manual labor to retrieve the right features for a given problem. With neural networks, data is "fed" into the model, extracting features without further human intervention and presenting a favorable result if the data has enough volume and adequate quality.

Artificial Neural Networks

On a very high level, Artificial Neural Networks (ANN) can be compared to the mammalian brain and nervous system. Neurons pass electric signals to each other. These signals travel from the axon terminals through the axon body, reach another cell and repeat the process through each neuron. One can imagine a path through the neurons where a signal travels through them. The neuron, the most fundamental unit, both receives and generates other signals. This way of thinking about ANNs might induce an error because it might lead to thinking that the signal remains stable as it travels through each neuron. Such might not be the case.

Abandoning the brain analogy, we can enter a fundamental concept of ANNs: the activation function. This is where the operations determine if a signal is passed forwards or not. When the input signal is small, then its output is 0. In case the value rises above a threshold value, a non-zero value is yielded. An ANN is composed of an input layer, a hidden layer, and an output layer. These nodes (or neurons) are linked with weight and threshold. When the output of a node surpasses its threshold, that node will be activated, and data will follow to the next layer. Case the threshold is of a lower value, the signal will not be transferred. During the training process, the weights will be corrected or adjusted according to how correct or incorrect the final output is compared to the real output. If an algorithm is meant to categorize a parasite in its ring stage, and the network results tell us it's a red blood cell. The network must receive a signal to adjust its weights responsible for the unfortunate outcome, diminishing their values and reinforcing the weights when the output is correct.

Therefore, the learning process of an ANN comprises the updating of the connection strength of a neuron. The distance between the predicted value and the actual value is calculated. According to this result, the weights in the network are adjusted to make this difference as minimal as possible. As more hidden layers are added to the network, more complexity is added. This is what is known as deep learning. There are several types of deep learning, but this work will only focus on convolutional neural networks (CNN).

Convolutional Neural Networks

CNNs excel at dealing with grid-like data such as images. Like ANNs, they also were inspired by a biological stance, but this time, it's specific to the visual cortex. Like them, CNNs, at first, extract different features from the image, like straight lines, curves, a particular color, or any other than encompasses an image. It goes from a hierarchy of features until, in the end, all are merged to make sense of all these features.

But why is there a need for a specialized deep learning network to deal with this kind of data? A typical neural network has a layer of nodes in line followed by a next one to which each node from the previous layer is connected. This follows for each layer in the network. While it's not impossible to train a standard ANN to deal with images, it's very impractical as the training time will get prolonged and probably result in overfitting. To deal with this, CNNs are sparingly connected, as shown in picture 2. If we take the example of the known CIFAR-10 dataset, one image is 32 x 32 x 3. This results in 3072 weights. It seems manageable at first, but as we progress in calculating this with more neurons, it will reach a point where such a large number of parameters will lead to the previously mentioned consequences. In a CNN, where not all nodes are connected, the number of parameters will be much lower.

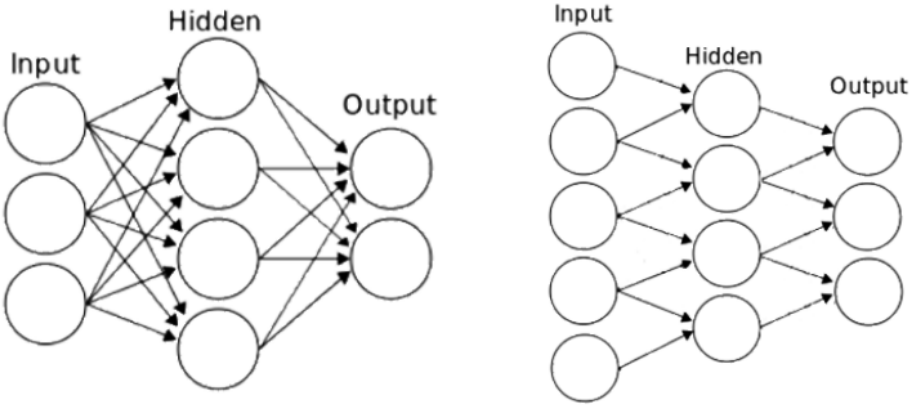

Figure 2 - Left: A typical ANN where each neuron is immediately connected to every neuron in the following line. Right: an example of how a CNN would be with sparse connections

2.2. BUILDINGS BLOCKS OF A CNN

A CNN architecture comprises several building blocks: convolutional layers, pooling layers, and fully connected layers.

1) Convolutional Layers

CNNs need input data, a kernel, and a feature map. If the input is an image with color, then it's a matrix of pixels in 3D and has height, width, and depth, corresponding to an RGB image. The feature detector, sometimes called kernel or filter, will iterate the receptive fields of the image, looking for the presence of a feature. Features here can refer to a specific trait of the image. At this stage, the CNN learns to recognize certain traits like colors, lines, circles, and other essential attributes that constitute the image at a low level. The kernel is a two-dimensional array of weights representing part of the image. There is a possibility of size variation, but the most common is a 3x3 matrix. This determines the size of the receptive field. The dot product is calculated between the input pixels and the kernel as applied to an image area. Then this dot product is passed on to an output array.

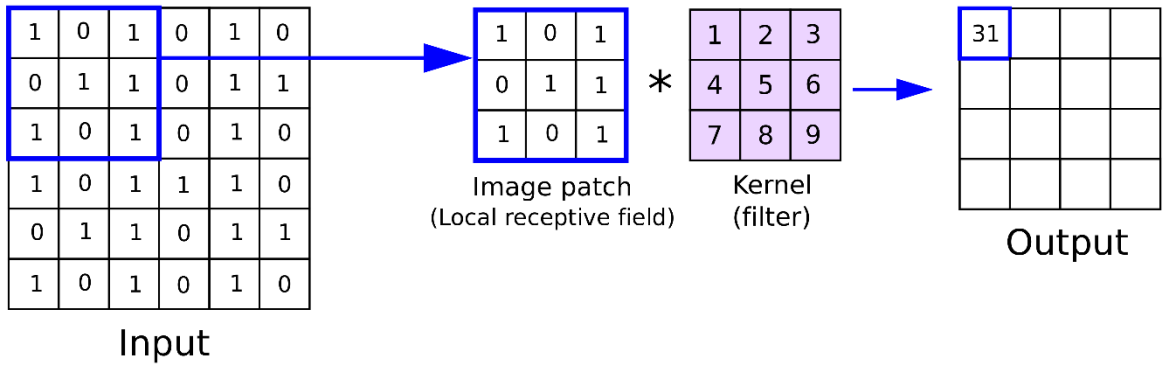

Figure 3: the process of a convolution. Source: Reynolds, 2019

The feature detector weights remain the same as it iterates through the image, another concept CNNs known as parameter sharing. Weight values adjust through the process of backpropagation and gradient descent. Three hyperparameters affect the volume size that must be defined before the training process.

- **Number of kernels**: this affects the depth of the output. For instance, five kernels result in five different feature maps and, thus, create a depth of five.
- **Stride**: The number of pixels the kernel moves over the input. A stride of two or more is uncommon; larger strides result in smaller output.
- **Zero Padding**: The usefulness of this hyperparameter is to be used when kernels do not fit the images, producing larger or equally sized output. Padding sets all elements outside of the input matrix to zero. There are three kinds of padding:
    - Valid Padding: Also called no padding. The last convolution is dropped if the dimensions do not align.
    - Full padding: Increases the size of the output by adding zeros to the border of the output.
    - Same Padding: Controls the size of the output layers, so it has the exact size of the input layer.

2) Activation functions

The need for activation functions surges from the linearity of the convolution layer operation. Without a non-linear activation function afterward, there will be no action, as neurons will not “fire” when needed, and learning will not be possible. Several activation functions are used according to their place and the design of the network. While ReLU is a typical function applied after a convolutional layer, sigmoid and softmax are used on the last layer. Sigmoid is used when the classification task is binary and softmax if the task is multi-class.

3) Pooling layer

Pooling layers, or downsampling, aim to reduce the number of parameters in the input. Just like convolutional layers, pooling involves a kernel across the entire input. The difference is that this filter does not have weights. It applies an aggregation function to the values in the receptive field, populating the output array. During this step, some information may be lost, but complexity is lowered, there is a lower risk of overfitting, and it improves efficiency.

There are max pooling and average pooling:

- **Average pooling:** The average value within the receptive field is calculated as the kernel moves across the image input. It is then sent to the output array.

- **Max Pooling:** As the filter iterates the input, the pixel with the maximum value is selected and sent to the output array.

4) Fully Connected Layer

The pixel values of the input image are not directly connected to the output layer. This is not the case in the fully connected layer. Each node is directly connected to a node in the previous layer in this step. The classification task is based on the features captured by the previous layer. FCL will use these features merged to attribute a final result.

2.3. CHOSEN MODELS

MobileNet

This network was proposed in 2017 by a team of Google. It has VGG-style blocks, yet it can reduce parameters and MACs vastly due to the substitution of the standard convolutional layer by the depthwise-separable convolution and pointwise convolution. Since the number of parameters suffers from a significant reduction, the model would be expected to lose descriptiveness and accuracy. However, this replacement by depthwise convolutions resulted in a 1% loss in top-1 accuracy. There are two hyperparameters in MobileNet defining its structure: width and resolution multiplier. The first,

α, controls the number of feature maps in each layer. Often, this hyperparameter is implemented so that the number of feature maps is divisible by 8, which is an advantage for GPUs memory structure. α can vary from 0 to 1. It has the effect of reducing computational cost by α^2. It reduces the computational cost and the number of parameters quadratically by α2 (Howet et al.,2017). The resolution multiplier, ρ, is responsible for the spatial resolution of the model's layers. P is chosen so that the input resolution of the network is 224, 192, 160, or 128. The model architecture can be seen in figure 4 below. All layers are followed by batch normalization and ReLU activation function.

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figure 4: MobileNet Architecture: Source: Howard et al., 2017

MobileNetV2

MobileNetV2 was released in 2018 and took on from the original MobileNet, but the ResNet architecture also inspires it. Its building blocks make use of bottlenecks and inverted residual connections. Linear bottlenecks are inspired by the importance of the number of channels of the layers. Channels serve to store and transform information. Information transformation requires more dimensions than storage. Therefore, these are architectures where the number of layers does not grow monotonously but instead grows and shrinks alternatively. That is how MobileNetV2 is constructed. The bottleneck layer includes three convolutional layers. A 1x1 pointwise convolution input feature in the first convolution is expanded to higher dimensionality. Then, these channels are transformed into a 3x3 depthwise-separable convolution, and another 1x1 convolution reduces the number of channels. This is the description of the bottleneck. There is no activation function like ReLU following this convolution. The expansion factor for the first expanding convolution must also be specified. This hyperparameter is fixed at 6. If the input holds X feature maps, then the first convolution will expand them to 6 X feature maps that the depthwise separable convolution will transform.

Another concept is the residual connections. These connect the input of a layer directly to the output, skipping layers of a block. The input of the residual block gets added to the output of the block, forming what is known as an inverted residual block. The residual connections connect bottleneck layers instead of expanded ones, which is typically done. Because input and output are summed up, dimensions must match.

MobileNetV2 has 3.4 million parameters and uses 300 million MACs. Compared to the first MobileNet, it used 20% fewer parameters and 47% fewer MACs. It has a 72% top-1 accuracy, higher than its predecessor.

NASNetMobile

NASNet stands for Neural Search Architecture (NAS) Network. So far, all networks discussed until now demanded manual labor and extended research. To achieve this, a lot of trial and error has undoubtedly taken time, and there may be situations where state-of-the-art techniques may not be suited. Also, there is no guarantee that an optimal solution was reached.

NAS automates network architecture engineering works by searching for the best algorithm to achieve the best performance on a specific task. It has three core components: search space, search strategy, and performance estimation strategy. Search space can be defined as the space of all possible architectures one could look for a given task.

There are two main types of search spaces: global search spaces and cell-based search spaces. The global search space has freedom for the arrangement of operations, allowing any architecture with elements such as convolutions pooling, dense layers, and global average pooling with different hyperparameters (number of kernels, kernel width, and height). It contains any repetitive patterns. The order of the layers is also considered.

The cell-based search space is focused on finding out the architecture of specific cells that can be combined to gather the whole network. This search is based on the principle that many manually designed architectures have repetitions. The advantage of this search space is that it yields smaller and more effective architectures that can also be turned into larger architectures.

As the name implies, the performance estimation strategy evaluates a neural network's performance from its design without the need to construct and train it. Every architecture returns a number that matches the performance or accuracy of the model when confronted with the test dataset with the given metrics. It can also help pass the results to the next iteration to produce a better model, or it can keep improving from scratch. It also has several possible search strategies:

- **Evolutionary algorithms:** Optimized algorithms that use mimicked biological mechanisms to find an optimal design within certain constraints. Through a process of evolution, it generates various components of neural network architectures.
- **Reinforcement Learning:** The first design of NAS (Zoph et al., 2017) involves a controller for proposing child models for evaluation. This controller is implemented as a Recurrent Neural Network (RNN). This controller network is used to sample from the search space with a

particular probability distribution. The output is a variable-length sequence of tokens used for configuring the architecture. The process will be iterated until convergence or timeout.

- **Grid and Random Search:** The grid search systematically screens the search space while the random search randomly chooses architectures to be tested. These approaches are solely indicated for small search spaces.

A score of 74.0% top-1 accuracy is achieved on ImageNet by the most miniature model of NASNet, which make it 3.1% higher than other hand-crafted architecture created to perform in mobile and embedded vision tasks.

EfficientNetB0

The Google Brain Team developed EfficientNet. Just like NASNetMobile, a neural architecture search was used. The idea of model scaling was applied. This is when scaling the model in terms of model depth, model width, and input image resolution is used to improve the model's performance and to scale the three to get better results, known as compound scaling. EfficientNetB0 is the most miniature model of the EfficientNets family, having 4 million parameters. It comprises seven inverted residual blocks using excitation blocks and swish activation functions. Each feature map channel was given an equal weightage produced by a convolution layer. Squeeze and excitation (SE) blocks are a method to provide weightage to each channel and not treat all equally. The activation function swish is a multiplication of a sigmoid and linear activation. The compound scaling is applied to scale it up from this baseline to obtain EfficientNets. Inverted residual blocks use depthwise separable convolution inside the residual block, which uses depthwise convolution first, followed by a pointwise convolution. This is key to reducing the number of trainable parameters.

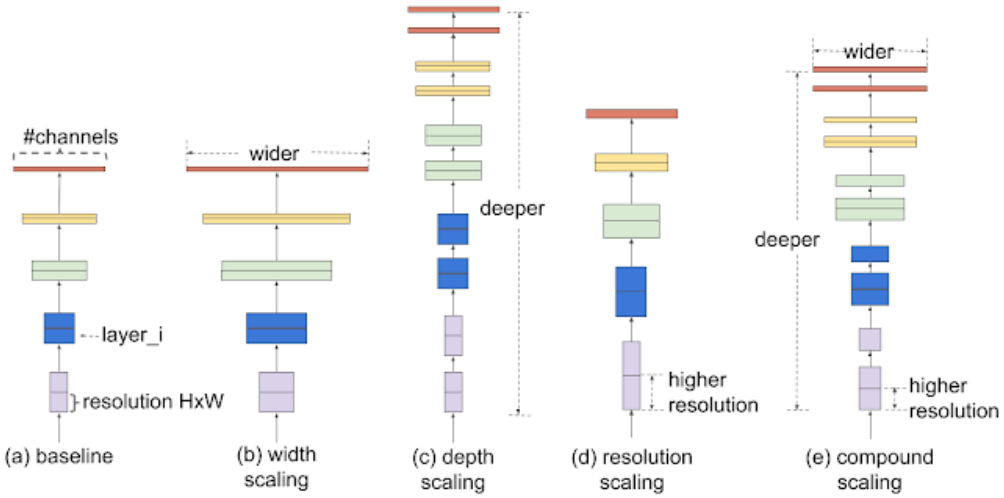

Figure 5- Comparison between different scaling methods. Source: Tan et al., 2019

Table 1 - Number of parameters, size, top-1-accuracy and top-5 accuracy for MobileNet, MobileNetv2, NASNetMobile and EfficientNetB0

Model	Parameters	Size (MB)	Top-1 Accuracy	Top-5 Accuracy
MobileNet	4.3M	16	70.4%	89.5%
MobileNetV2	3.5M	14	71.3%	90.1%
NASNetMobile	5.3M	23	74.4%	91.9%
EfficientNetB0	5.3M	29	77.1%	93.3%

2.4. OF SMALL DATASETS AND TRANSFER LEARNING

One thing about deep learning is that it requires large amounts of data to work correctly and learn in such a fashion to return accurate results. In an ideal scenario, a dataset has a substantial volume, veracity, annotations, and reusability. An ideal dataset is Findable, Accessible, Interoperable, and Reusable (FAIR). The ground truth must be accurate. These scenarios are often not the case in the medical field. Regarding medical imaging datasets, open-source large-quality datasets are still in low numbers. One of the main reasons is privacy. Standards differ vastly from country to country, but broadly speaking, laws act in the ordinary citizen's interest to protect their health data. Data is usually associated with sensitive patient data such as social security numbers and other private information. Before proceeding to create any dataset, anonymization is required. Another issue for supervised learning is the manual labor required to have all samples attached to its label. Annotations can also typically be done only once by the same expert or group of experts. This is a problem since sometimes different professionals can differ in opinion regarding the same image in terms of diagnosis. So, datasets can show significant variation. These issues and others surrounding them make it challenging to gather large datasets of medical data.

One tactic to counteract the issue of small datasets is transfer learning. With transfer learning, the network weights are not initialized from scratch. The model will be initialized with weights from a CNN trained on a larger dataset. A popular dataset where many networks, including the ones used in this work, are trained is ImageNet. This dataset contains 1000 classes of natural images such as animals, trees, houses, and many others familiar to the day-to-basis. It does not have any sort of medical imaging. It is also the goal of this work to perceive how well the chosen models can adjust, in

conjunction with the transfer learning technique, such as to classify an unseen class like the malaria parasite. There are several ways to use transfer learning with CNNs:

- **As a feature extractor:** We use a CNN pretrained on ImageNet, remove the fully connected layer at the end, and use the rest of the network as a feature extractor for the new dataset. SVM or kNNs can also be used.

- **Fine-Tuning:** This strategy removes the fully connected layer and fine-tunes the weights of the pretrained network by continuing the backpropagation. We can finetune all the layers, or we can keep some layers fixed and only fine-tune some of the higher portions of the network. The earlier features of the CNN have more generic features that can be useful for many tasks. Successor layers become more specific to details of the classes of the original dataset. For instance, if ImageNet contains a class of trees, then those features can be helpful in a new dataset where the task is to classify different kinds of trees. This technique avoids overfitting by fixing weights on the first layers, which have more general features.

3. LITERATURE REVIEW

This chapter reviews the importance that microscopy represents as a malaria diagnosis, and its advantages and disadvantages are weighted and compared to other methods. Machine learning and CNNs are presented as potential solutions to automate the diagnosis when combined with mobile phones as tools to capacitate microscopy further.

3.1. MICROSCOPY AS DIAGNOSIS OF MALARIA

The first symptom that raises suspicions of malaria infection is fever or a recent history of fever. However, it is not always the right indicator, as fever is also common in malaria-endemic areas for other reasons. Since malaria's symptoms do not identify the disease by themselves without the need for medical confirmation, this results in many false positives, with different conditions being overlooked and drug administration being wrongly administered, contributing to the problem of drug resistance. Parasitological confirmation is necessary not only for case management but also for the comprehension of the malaria burden.

Microscopy of Giemsa-stained blood films remains the golden standard when it comes to methods of malaria diagnosis. Since 2019 WHO has recommended that suspicions of malaria should be confirmed by microscopy or RDTs. A small blood sample from the patient's finger is collected and spread as a thick or thin blood smear. It is then stained with Giemsa stain (others may still apply, but this is the most common) and examined under a 100X oil immersion objective.

It has high sensitivity and specificity, detecting as low as 50 parasites /µL blood even under field conditions. It can determine parasite density, distinguish between parasite life cycles, and differentiate between parasite species and other diseases. Yet, its accuracy and usefulness depend on the quality of materials involved, quality control, and the experience of the microscopists. It is challenging to maintain these requisites, especially in peripheral health services, just where malaria has a high prevalence. Another factor impeding this method of reaching remote places is the lack of electricity. Due to its versatility in results when properly applied, it is justified why it has been recommended and widely used. However, there is a need to project this method further by mitigating its disadvantages. One way to do this is to automate the results by reading and interpreting the received image input.

There are other methods of diagnosis, such as Polymerase Chain Reaction (PCR) and Rapid Diagnostic Tests (RTDs), to name a few examples. However, molecular-based methods require highly trained personnel and expensive equipment and are time-consuming, while immunology-based methods, for instance, cannot differentiate between past and present infections. RTDs also suffer from this disadvantage and cannot quantify parasite density.

3.2. MOBILE PHONES AS MEDICAL AIDS

In the previous section, the pros and cons of several methods of malaria diagnosis were seen, as well as its pertinence and availability and adaption in remote places. The most appropriate diagnosis, if based on portability and reliability, is microscope based. However, its shortcomings, like the need for personnel with expertise and the electricity demand, can be resolved thanks to new and emergent technology. This is particularly true when it comes to lighter microscopes and the development of

mobile phones. The automation of diagnosis will decrease the number of microscopists needed to travel to remote areas with the guarantee of a reliable diagnosis, easy to carry, and cheap equipment that will last many hours without recharge.

The ownage and use of mobile phones keep on growing. It is estimated that, in 2015, there were over 6 billion cell phone subscriptions worldwide, with 75% of the population having access to mobile phone networks (Zhu et al., 2011). This includes low-resource settings. Due to this immense adherence, prices remain low cost even with constant advances in both software and hardware. One of these significant advances is the advanced camera features which make them an ideal platform in many advanced imaging ad mHealth applications, which result in portable field-of-care (POC) devices (Zhu et al., 2011). This is of extreme relevance for clinics and remote places in low-resource settings where healthcare infrastructures are often precarious. Many papers are dedicated to exploring mobile devices for low-cost alternatives to conventional microscopy techniques, including malaria detection (Pirnstill et al., 2015). A practical approach is to attach the smartphone to the eyepiece of a microscope with an adapter.

3.3. THE APPLICATION OF MACHINE LEARNING IN MALARIA DETECTION

This section is dedicated to an overview of some of the previous work on detecting malaria with the previously discussed method of diagnosis. It includes studies that focus solely on the performance of the algorithms and some that focus on the trade-off performance vs. size of the final model. This last factor is crucial if we want to deploy the model in a device with low computing power. Section 3.3.1 presents some applications of machine learning algorithms. Section 3.3.2 is dedicated to the study of CNNs in the field, and the 3.3.3 section pertains to an application of quantization and evaluation of performance when deployed.

3.3.1. Of supervised and unsupervised methods

There have been numerous experiments on the topic of this work beyond CNNs. Monhaty et al. have tested an auto-encoder and Self-Organizing Networks (SOM). Another important task when dealing with cell images is segmentation. Their experiments led by segmenting and classifying cells as parasitized or uninfected. With SOM, each blood cell is isolated from other cells based on minimum area. For classification, color, texture, and shape were extracted as features. The spatial distance between cells is calculated with the result for accuracy being 79%, 80%s sensitivity, and specificity of 78%. As for the Auto-Encoder, it also focuses on feature extraction. It starts with 12500 input nodes and ends with a total of 30 valuable features. It had an accuracy of 87,5%, a sensitivity of 84%, and a specificity of 80%. (Monhaty et al.,2019)

Malhotra et al. Compare a CNN with a k-Nearest Neighbour (kNN) and outweigh the advantages and disadvantages of both methods. kNNs are easier to implement: to predict a class, the one with the most occurrence within kNNs must be found. Since it is based on the calculation of distances, a good distance function may be hard to find. CNNs extract features automatically and are more robust to noisy data, but they depend on good enough hardware to perform. The result for kNN was 74,7% in accuracy, and the CNN got 94.56%. (Malhotra et al, 2020).

Li et al. focused on not a binary classification problem as seen so far. Instead, their focus was the distinction between the multistage life cycles of the Plasmodium vivax parasite. A deep transfer graph convolutional network (DTGCN) was generated for that purpose. The combination of CNNs for feature learning, source transfer graph building, and unsupervised GCN resulted in a promising method of diagnosis. To prove the capabilities of this method and knowing that datasets are often scarce with few samples, the authors have used smaller subsets of the original dataset and compared the achieved results in accuracy. With 20% of the data, an accuracy of 89.3% is achieved. When 40% of the dataset is used, the score rises to 97.7%, with the same score as 60% of the dataset. (Li et al.,2021)

Islam et al. proposed a multiheaded attention-based transformer with outstanding scores of 99.25%, 99.00%, 99.50%, and 99.99% for accuracy, precision, recall, and an AUC%, respectively. To further comprehend what the model was learning, Grad-Cam was used. (Islam et al., 2022)

Bibin et al. have experimented with a deep belief network (DBN). 4100 images of parasitized and non-parasitized peripheral blood smears were used for this purpose. The DBN was pretrained utilizing contrastive divergence technique for pre-training to stack restricted Boltzmann machines. Features from the images were extracted to train the DBN and start the DBN's visible variables. Texture and concentrated color were used as a feature vector. In the end, a backpropagation that calculates the probability of class labels was employed to fine-tune the DBN. The best size of the architecture was "484-600-600-600-600-2". The scores were 89.66%, 97,60%, and 95,92% for F-Score, F-Score, sensitivity, and specificity, respectively. (Bibin et al., 2017)

Linder et al. have focused on the detection of ring-stage falciparum parasites. Extracted image features include local binary patterns, local contrast, and Scale-invariant feature transform descriptors. These were then transmitted to a support vector machine (SVM) classifier. This technique has reached a sensitivity of 95% and a specificity of 100%. To assess how efficient this system is, two expert microscopists participated by manually deliberating if a cell was infected or not, independently from the algorithm. The correlation coefficient between the manual and automated methods was 0,97. (Linder et al., 2014)

3.3.2. CNNs

Many examples in the literature of transfer learning are applied to classifying malaria in blood smears. VGG16, AlexNet, NASNetMobile, Inception, Xception, and ResNet5 were used in the same task with an imbalanced dataset of 4500 infected and 2500 uninfected (Sriporn et al.,2020). Images were rotated using data augmentation in different degrees to increase the number of samples and tackle the imbalance issue. RMSprop, SGD, and Nadam were tested, as well as ReLU and Mish's activation functions. The model that turned out to be the best was Xception with Nadam optimizer and Mish activation function, with an accuracy of 98.8%. (Sriporn et al.,2020).

Using the original version of the dataset used in this work, Rajaraman et al. conducted an experiment where a custom CNN composed of 3 convolution layers with a max-pooling layer after each layer, two fully connected layers, and ReLU activation function was compared against pre-trained models. These models include VGG16, AlexNet, ResNet50, DenseNet121 and Xception. The results place the custom CNN trained from scratch with an accuracy of 92,7% and the best pre-trained model with 95,9% and a sensitivity of 94.7% (Rajaraman et al., 2018).

An ensemble learning with pretrained VGG19, SqueezeNet, InceptionResNet-v2, and a custom-made CNN was proposed (Rajaraman et al., 2019). The combination of features of these four models was put together to discriminate between healthy and malaria-infected cells. All models were also trained separately. VGG19 outperformed all the other models, including the ensemble, with an accuracy of 99.32% and a sensitivity of 99.31%. The ensemble reached 99.11% accuracy and a sensitivity of 98.94%. (Rajaraman et al., 2019)

Generally, most studies focus on achieving a higher accuracy using several deep learning methods. There are exceptions to this, such as Quinn et al. and Rosado et al., who took the problem of efficiency on top of good performance. Unfortunately for both, accuracy suffered a drop in exchange for lighter models. The latter also tackled the use of mobile phones as means to detect malaria and white blood cells (Rosado et al., 2016). As for their scores, the sensitivity was 80,5%, and specificity was 93.8% for trophozoites and 98,2%, and 72,1% for white blood cells. SVMs were used with a sensitivity and specificity of 98.2% and 72.1%, respectively, for white blood cells. Rosado et al. also showed that their model could be deployed into cell pho. However, they were not low-cost items more prevalent in poorer regions. (Rosado et al., 2016).

Fuhad et al. have presented a work to tackle these issues. Several experiments were carried out: distillation and autoencoder training with a comparison of 10 models. The best-performing model resulted in an accuracy of 99.5% using an Autoencoder method. In terms of results, it is comparable to the work of Rajaraman et al. The achieved model requires only 4600 flops which is suitable to be deployed. (Fuhad et al., 2020)

Not only are models important, but data pre-processing and data augmentation can also play a significant role in accuracy. Rahman et al. have shown that standardization, normalization, and stain normalization do not contribute to a better result; instead, data augmentation shows promising results. Their work includes horizontal and vertical flips, gaussian, blur, rotation, horizontal and vertical shifting, darkening, and lightening, ZCA whitening, and feature-wise standardization. A customized version of VGG16 was employed, and a custom CNN with an SVM as a classifier. The named TL-VGG achieved an accuracy of 97.77%. (Rahman et al., 2019)

Militante et al. have focused on the three versions of the MobileNets. MobileNetV3 achieved the highest score, with 96.5% accuracy. MobileNetV2 performed the worst; this last result coincides with the outcome of this work. (Militante et al., 2021)

Montalbo et al. have employed EfficientNetB0 with different ending layers via fine-tuning. In their work, this EfficientNetB0 has surpassed models like NASNetMobile, InceptionV3, InceptionResNetV2, and ResNetV2-152. Although only NASNetMobile has been used in this work out of all these models, the result is similar, as EfficientNetB0 was the highest-achieving model in terms of performance. Montalbo et al. achieved a score of 94.70% for 50 epochs. A clear indicator of the potential of lighter-weight models for classifying malaria. (Montalbo et al., 2021)

Arshad et al. have created a new dataset and started a cascaded CNN with the goal of, in the first instance, differentiating between healthy or infected cells. Only then will the lifecycle be detected. To localize the cells and keep the computer power low, a watershed is used to localize the cells. The mean accuracy was 79.61% and 82.04%. The results show that the cascade form is more efficient than a one-stage model in detecting lifecycles solely. The future is to implement the algorithm in an app to evaluate its performance while deployed. (Arshad et al., 2022)

3.3.3. Going further: methods of compression

So far, we have seen methods where it's entirely possible to deploy a network in a small device to techniques that give excellent results but are not realistic when using them. It is a transversal problem within machine learning that has been studied. Although there are several different techniques, the goal converges towards diminishing the original size of the network.

Revisiting a bit of the concept of neural networks, these are made of weights and biases, also referred to as parameters. Each neuron has its activation function. The parameters and activations are what get tuned during the training stage. Of all these values of parameters and activations, the majority gets stored in memory. The typical standard is 32-bit floating-point representation values. This allows for high precision and accuracy for the neural network. So, in sum, for a network with millions of parameters and activations, information being stored as a 32-bit value, memory usage quickly adds up. ResNet architecture contains 26 million parameters and 16 million activations. Adding this up to the 32-bit floating point values, the network requires 168 MB of storage. One of the ways to reduce the memory storage needs is to quantize the model.

Eze et al. employed this method to classify malaria. They evaluate the performance of a custom-made CNN, a VGG19 version with some frozen layers, VGG19 with fine-tuning, and MobileNetV2, employing different quantization methods. They compared the untouched version of the networks with INT8WO compressed weights, Float16WOnly and Float16WAndA. Both performance and memory were accessed in terms of trade-offs. The results were auspicious, with MobileNetV2 reaching 96% to 98% between all methods. The authors claim that if only one method could be chosen to be deployed, then it would be the custom CNN with 95% precision and 99% precision in the best average across all modalities of all models. (Eze et al.,2021)

Quantization is not the only way to make a network lighter in its size. This work is also dedicated to a technique known as pruning. It's no secret that not every weight contributes the same way to the network. Some may not even contribute at all. These weights are not worth keeping in the model, for they only occupy space having zero retribution. To nullify them, these weights are set to zero. To use the correct terminology: we make the model "sparse." In this work, magnitude pruning is used: it scores the weights according to their overall importance and contribution, and the rest, below a certain threshold, is discarded. Since pruning happens during training, the weights that matter can adjust to the impact of the weights that disappear. In this way, the model remains robust against the sparsity of the fly. The TensorFlow Model Optimization toolkit offers a simple yet effective implementation. There are two main approaches: constant sparsity kept constant during training, and polynomial decay, which means that the degree of sparsity is changed during training. More about it is in section 4.2.4, where its implementation is done.

4. METHODOLOGY

This chapter will elucidate details about the experiments made during this work. It contains information about the used dataset, discusses the importance of data augmentation and if it might be favorable in all cases, describes the metrics considered necessary for a medical field classification task, describes the process for each experiment, including methods of transfer learning, hyperparameters and pruning with different levels of sparsity.

4.1. DATASET

A publicly available dataset released by the National Institute of Health (NIH) was chosen as it is readily available and has been heavily used for malaria detection. It contains thin blood smear films of 150 P. Falciparum and 50 healthy patients prepared using Giemsa staining solution to enhance the visibility of the falciparum parasite. The images were photographed at Chittagong Medical College Hospital in Bangladesh, and expert slide readers manually labeled all at the Mahidol-Oxford Tropical Medicine Research Unit in Bangkok. The original dataset is balanced and contains 27,558 cells, with 13,779 parasitized cells and 13,779 uninfected cells. Images have different color distributions due to varying bloodstains during data acquisition. They also contain impurities and artifacts, making the task more challenging. Images are not the same size, with the minimum being 46 x 46 and the maximum 385 x 495, hence the importance of resizing.

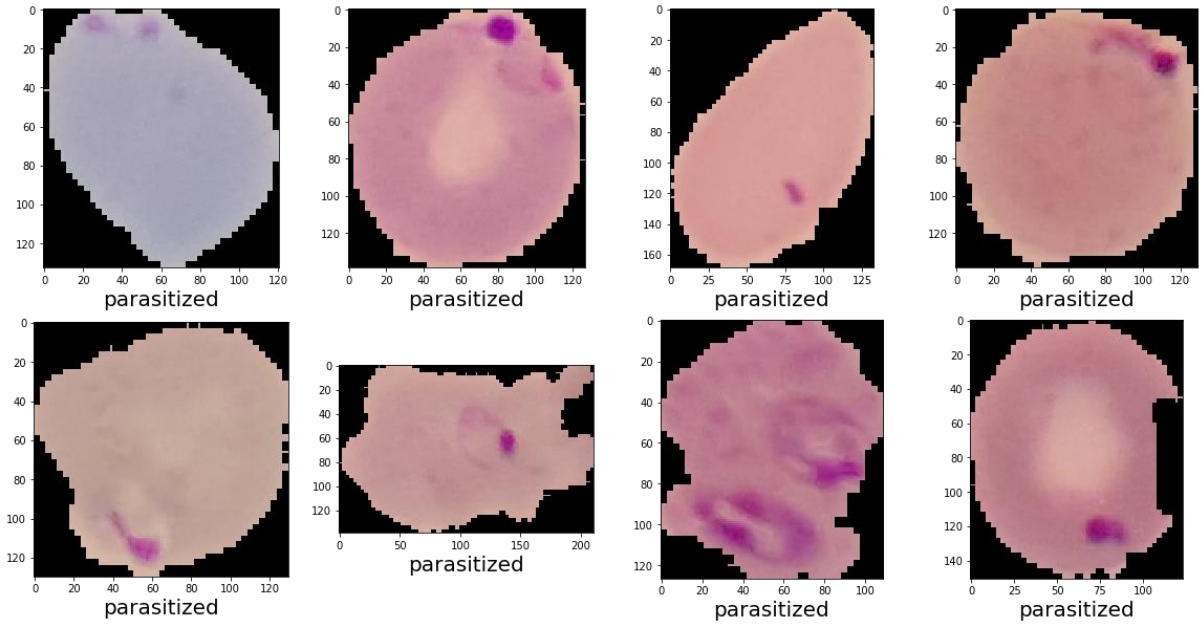

Figure 5 - Samples of cells infected with the Falciparum parasite. Although there are slight color differences, the more salient purple circles, denote the presence of the parasite.

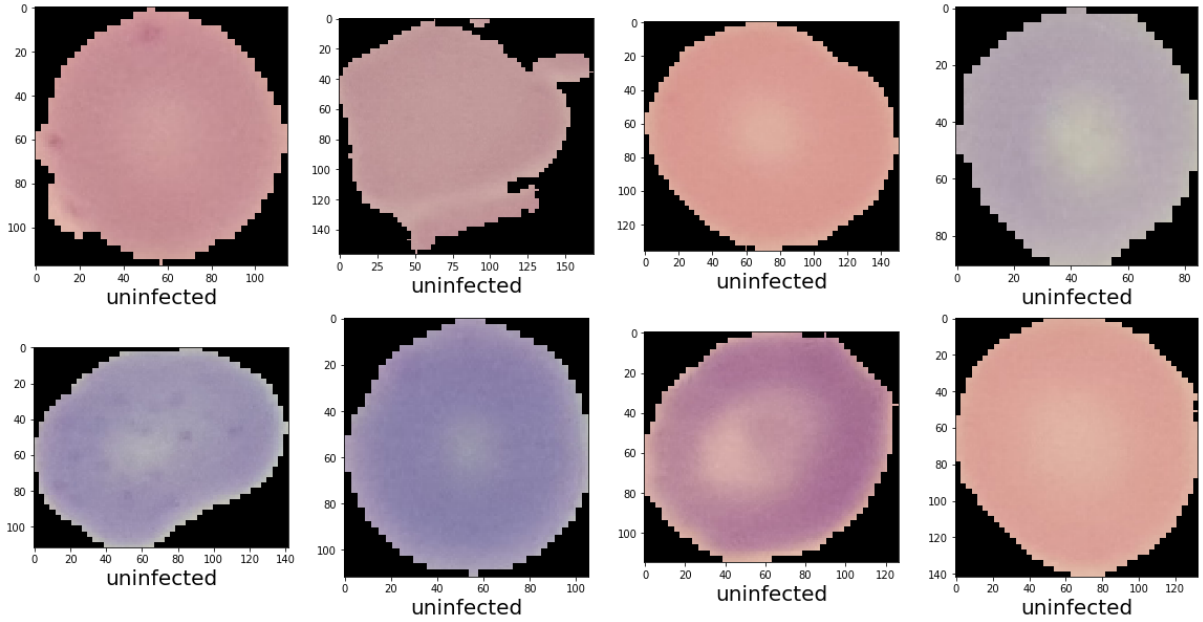

Figure 6 - Examples of healthy cells. The surface is clean and does not present any anomalies.

However, Fuhad et al., during their research, noticed that some samples were mislabelled. It can be seen in figure 5 that parasitized cells, regardless of color differences, show a smaller rounded shape with pronounced contours. This was not the case with some of the falsely labeled images as positive and negative. Fuhad et al. had an expert analyze the images thought to be wrong. As a result, 647 images mistaken as positive were removed, leaving 13,132 true parasitized images, and 750 falsely images annotated as uninfected were also discarded. The number of true uninfected is 13,029. The resemblance of the falsely labeled samples with their counterparts can be seen in figures 7 and 8.

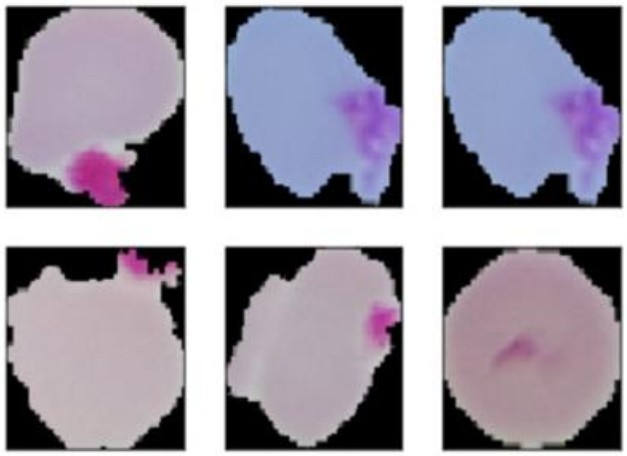

Figure 7 - Falsely labeled as uninfected. Source: Fuhad et al., 2020

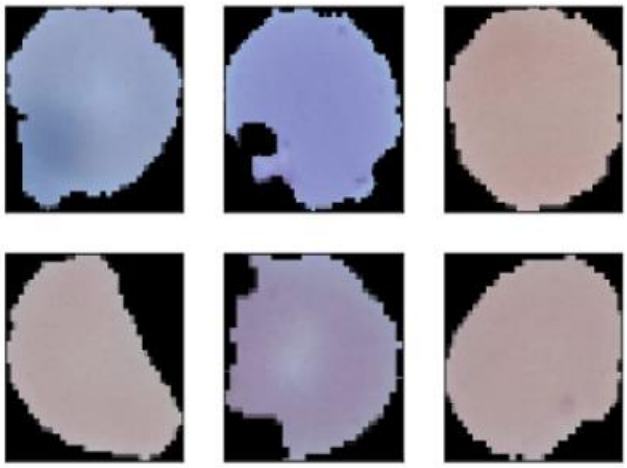

Figure 8 - Samples falsely labeled as parasitized. Source: Fuhad et al., 2020

The dataset was divided into training, validation, and test sets with the specific values of 70%, 20%, and 10% for each, respectively, ending with a training set with 18312 images and a validation set with 5231, and a test set with 2618. The number distribution can be seen in table 2.

Set	Number of Images	%
Training	18312	70
Validation	5231	20
Test	2618	10

Table 2 - Dataset distribution for training, validation, and test sets

4.2. FLOW AND CONSTRUCTION OF THE ALGORITHM

Training models from scratch involves a lot of trial and error and can be very time-consuming. Therefore, transfer learning with pretrained models is a very enticing method of achieving faster and excellent results in many modalities. It was seen in section 3.3.2 that custom-made CNNs had success on this dataset and attained high accuracy. However, adopting a whole new model from scratch may be a long process, and it is usually not stated how long the authors took to get the model to work correctly.

It is the premise of this work that pretrained models can achieve high accuracy, recall, precision, and F1-Score for the task of malaria detection. Mainly due to the reasons stated in chapter 3, such as the necessity of models being small enough to be deployed in smaller devices with less computer power, the chosen models must fit the criteria of having an architecture built for this exact purpose. Therefore, the selected models previously described in chapter 2.3: MobileNet, MobileNetV2, EfficientNetB0, and NASNetMobile.

While lightweight, these models can achieve peak performance as more parameters do not translate into better performance. However, we must remember the “no free lunch theorem.” No model is ideal for all contexts and datasets. Besides, hyperparameters are an essential part of deep learning that dictate how a model will behave during the training phase and, consequently, on prediction ability.
The second premise of this work is that the compression pruning method can keep the model's initial performance while simultaneously decreasing its size.

For that matter, and to achieve a conclusion regarding the assumptions, the following steps were taken:

1) All models were trained using the two transfer learning methods described in section 2.4.
2) The models with and without data augmentation on both transfer learning methods.
3) The best model was chosen for pruning. Both constant sparsity and polynomial decay were tested with different final sparsity of 50%, 70%, and 90%. Details are given in section 4.2.4.

All of the experiments rely on open-source software and platforms:

- **Python**: An object-oriented, high-level, general-purpose language commonly used in data science, automation, and software development. Python supports modules and packages, encouraging program modularity and reuse of code.
- **Tensorflow**: Developed by researchers on the Google Brain team. It is an end-to-end open-source platform for machine and deep learning. It contains an ecosystem of tools, libraries, and overall resources for the construction and research of artificial intelligence
- **Keras:** an open-source high-level neural network library written in Python. It was created to be user-friendly, extensible, and modular to facilitate experimentation with deep learning networks.
- **Kaggle**: a platform known for hosting data science and machine learning competitions. Besides, they offer users the opportunity to publish and search for datasets to use in private or public projects for free. All examples in this work were run using Kaggle’s platform using an NVidia K80 GPU.

4.2.1. Data Augmentation

Although plenty of research shows that data augmentation is a distinctive form of tackling the issue of small datasets, it is equally important to pay attention to the transformations to be done. Each must be done according to the type of dataset one has so as not to distort the class's original shape to something unrealistic. For instance, Elgendi et al. examined several geometrical transformations on a dataset of chest X-rays with covid patients. Three combinations of data augmentations were compared against each other, and one with no augmentation at all. Surprisingly, the one with the best performance was precisely the latter. The authors argue that this might be due to transformations that lead to unrealistic images that would otherwise be impossible to see in the real world. For instance, flipping the x-ray images on the vertical and horizontal axis creates the wrong positions of the heart and lungs (in the vertical case). Rotations have the same effect as switching the image’s original axis.
In the case of malaria, basic geometrical transformations like horizontal and vertical flips, for instance, would not affect the images to an unrealistic extreme. Lighting experiments might be more suitable if

they were the goal of this study. The models will be tested with data augmentation and without it to see if it benefits performance and training timing.
Table 3 represents the modifications made in terms of geometrical augmentations.

Augmentation	Value
Rotation range	10
Zoom Range	0.2
Horizontal Flip	True
Vertical Flip	True
Rescale	1/255
Resize	128x128 / 224x 224

Table 3 - Data augmentation transformations applied to the dataset

Another important action is to rescale the images in the range of 0-1 as well resizing the images. Neural networks receive inputs that are the same size. When the fixed size is larger, then less shrinking is required. This means that the images will suffer less deformation of their features, thus, mitigating the accuracy degeneration because of said deformations. If images are larger, they occupy more memory space and cause a larger CNN. The choice of the fixed size is a trade-off between accuracy and computational efficiency. The chosen image fixed size for this work is 128 x 128 for all models except for NASNetMobile, which requires a size of 224 x 224. All of these actions were performed under the Keras Image generator class, allowing all transformations on the fly.

4.2.2. Applied Transfer Learning

All pretrained models were trained on ImageNet. ImageNet contains 1000 classes, all natural objects and none with medical images.

As stated previously, the dataset used in this work contains only two categories: parasitized or uninfected, making it a binary problem. For all models, the last layer is removed as its pragmatical goal is to classify one of the 1000 images of the ImageNet dataset, and we only have two classes. A dense layer will always replace the final layer with a neuron and a sigmoid activation function. This must be done as, out of the 1000 classes of ImageNet, none belongs to our dataset where the models will be trained. The following experiments are as follows:

1) The last dense layer has two added layers before it: a global average pooling and a dropout of 50% with the intent to regularize the network. Values of 20%, 50%, and 70% were tested, and it cleared that the value of 50 was the one that worked best in terms of leading to faster convergence and preventing overfitting.
2) An experiment where all models had the last three and, in a third experiment, four layers frozen, preventing them from having their weights modified. This will speed up the training

time of the models. Here, as in the above cases, there is no guarantee of the number of layers to freeze to get the best results. This is a trial-and-error experiment. After that, using the Sequential API, a new classifier was added, plus dropout as a regularizer. Figure 9, below, represents the following layers that were added to the models in detail and by descendant order.

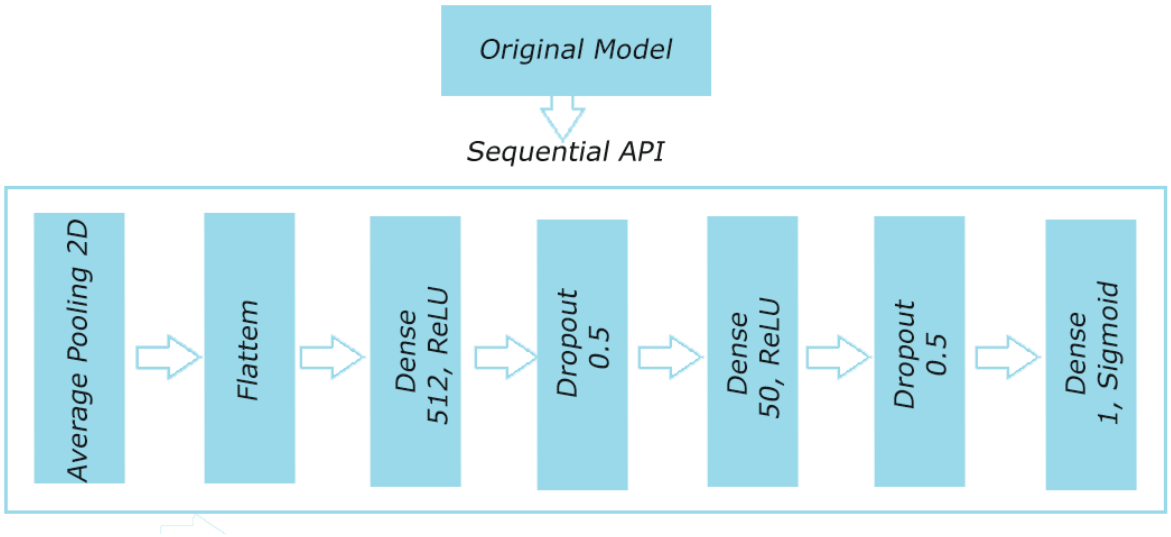

Figure 9: Illustration of experiment number 2 with the new classifier and precedent layers

4.2.3. Hyperparameters and optimization

Learning Rate and Optimizer

The learning rate is probably the most essential hyperparameter in deep learning. Just like when fine-tuning an architecture, there is no given rule to choose the ideal number. The learning rate controls the size of the step that an optimizer takes to reach the minimum of the loss function. If it is too large, then there is a chance that the model will converge too quickly to a suboptimal solution; when it is too small, then it can cause the process to get stuck at a local minimum.

Thankfully, Keras comes equipped with callbacks that allow us to modify the learning rate instead of remaining stagnant and not achieving faster convergence or higher accuracy. The callback used to alter the learning rate was Reduce on Plateau. In all cases, the starting learning rate is 0.001. As the name implies, the role of this callback is to reduce the learning rate as a metric to stop improving. When validation loss (the chosen metric for the callback) has stopped decreasing, the learning rate will decrease. Hopefully, this will result at the end of the staleness of the algorithm. Validation loss may represent a better choice to be monitored for this case. As discussed in section 4.2.5, accuracy may not be ideal for a medical setting. With the validation loss, there is more realism and precision as it lets us know the true rate between the real outputs and the predicted outputs.

This callback allows, through arguments, to specify the number of epochs to count from where improvement has stopped and the value to be considered as stuff. Adding to that, there is the factor to which the learning rate will be actualized. In table 4, below, are the definitions used in the arguments for this work.

Arguments	Value
Monitor	Validation Loss
Factor	0.2
Patience	2
Minimum Delta	1e-7

Table 4 - Data augmentation transformations applied to the dataset

Equally important is the choice of the optimizer. The function of this is to minimize the loss function. Adam is an optimizer of reference that works well on many problems, with the perks of taking the best of RMS-Prop and momentum and combining them. However, it has limitations: it can suffer from a weight decay problem and does not converge to an optimal solution in some areas. (Chen, 2020). To address these issues, new variants have been proposed and studied recently. Nadam can be seen as an extension of Adam but with Nesterov momentum, which takes a decaying average of past steps and steps in that direction first. Then the gradient is computed from that new position using the data, performing a correction. Then the weights are updated twice each iteration, using momentum and the gradient algorithm.

Augmentation	Value
Initial Learning rate	0.001
Optimizer	Nadam
Batch size	32
Maximum Epochs	30

Table 5 - Data augmentation transformations applied to the dataset

Stopping Overfitting

Overfitting happens when a model stops to learn and, instead, it starts to “memorize.” After that, it will not be able to generalize when confronted with the test set. The model may be too complex for the data or train for too long. High variance and a low error rate are a signal of overfitting. In CNN, visualization of this factor is critical, hence the importance of the validation set. The contrast between the loss and accuracy against the train set can tell us when and if it starts to happen. Keras offers easy-to-implement callbacks to counteract this issue.

Early stopping allows to stop of the training process before the originally defined number of epochs is reached when the validation loss comes to its lowest point. However, one can falsely adjust the

callback to finish too soon because the model might worsen before improving again. To add a delay to the number of epochs, the callback waits until stopping; there is the patience argument. We can also choose the minimum change in validation loss or accuracy to qualify as a better score through the argument of the minimum delta.

Model Checkpoints saves a snapshot during training every time it reaches a new and better score. Not only it saves progress in case of a crash, but it keeps the best performance even if then it starts decaying

4.2.4. Pruning

In section 3.3.3, the concept of magnitude pruning was reviewed, and there are two ways to reduce the meaningless weights to zero in Tensorflow. Applying pruning to a TensorFlow model must be done using a pruning schedule.

This wrapper lets Tensorflow know how the pruning will happen: is it constant sparsity, or does pruning occur within a schedule? This introduces information about whether a layer should be pruned at a particular pruning step and, if so, what sparsity it must be pruned for. Two ways this can be achieved are through constant sparsity or polynomial decay. The main difference is that for the latter, the pruning schedule induces a sparsity level built upon a polynomial function from one sparsity to another. In contrast, constant sparsity remains stagnant when it prunes a layer. To have control of these factors, Keras API provides arguments.

Constant Sparsity pruning Arguments

One can set pruning arguments based on our needs using those options. We can control the sparsity level used when necessary by setting a consistent target sparsity. Both the *begin step* and the *end step*, as well as the frequency, determine the latter.

Pruning can be applied to the entire training process, the initial part, or other configurations. You can configure begin a step to apply pruning only to the last stage of training. Frequency can be used to set how frequently pruning must occur.

Table 6 - Constant Sparsity applied arguments

Arguments	Value
Target Sparsity	50/70/90
Begin Step	20
End Step	(Num Images/ Batch size) * number of epochs
Frequency	100(default)

One risk in applying the same callbacks with the pruning-related hyperparameters in constant sparsity is setting an ending step that completes pruning by an epoch that is not reached by the time the

callbacks interrupt the training stage. For instance, and for the purpose of this work, this is achieved by dividing the number of images by the batch size and multiplying this result by the number of epochs for the mole to be pruned. If the model stops improving by epoch 9, then it won't have been thoroughly pruned as defined in the arguments.

Polynomial decay-based pruning

Using **polynomial decay-based sparsity**, more or less sparsity can be used with increasing or decreasing speed as training progresses. Here, the user must provide an *initial sparsity* and a *final sparsity* percentage. In this case, the user must supply both an initial and a final sparsity percentage. A begin and end step, as well as a frequency, must be handed down, similar to constant sparsity. The power argument is new, which symbolizes the exponent of the polynomial function to be applied in the sparsity calculation.

Arguments	Value
Initial Sparsity	20
Final Sparsity	50,70,90
End Step	(Num Images/ Batch size) * number of epochs
Frequency	100

One might think that any of these methods will actually "cut" the weights out of the network, and thus, we could afterward see the model with fewer parameters compared to its previous version. In Keras, a low-valued weight can be "masked" with zeros, and the number of non-trainable parameters may increase. Although, in some cases, comparing before and after pruning can be made with other methods, it is not as significant here. If the number of parameters is checked before applying any magnitude pruning as provided by Keras, they will be the same as the pruned model at the end of the process.

4.2.5. Metrics

A classification metric is a measure of the performance used to evaluate the model's performance when it comes to assigning observations to specific classes. The adequate one depends solely on the kind of problem we have. The metrics used in this work and their pertinence to the medical field and for the case of malaria detection are now discussed.

Since our problem has only two classes, binary accuracy is more appropriate than accuracy, which is commonly used for multiclass classification. It usually describes how the model behaves when it comes to each class. Suitable for when all classes have the same importance and the balance of the dataset is equal. Its calculation is the ratio between the number of correct predictions and the total number of

predictions. Accuracy of around 50% is not better than random guessing. It also is not very informative when letting us know where the model makes the most mistakes. Regarding health classification problems, it's not the ideal metric as it's very useful to know the false positives and negatives instead.

Recall focuses on positive samples. It is a measure of how many relevant elements were detected. In our dataset, we only have parasitized cells and uninfected ones. In this case, recall considers the parasitized cells (positive) compared to the overall number of parasitized cells.

There is a trade between trying to detect all relevant items and avoiding making wrong predictions. In the malaria case, false negatives should be avoided since they can have lethal consequences. Recall can be here more adequate than precision in this case.

Precision is the ratio between true positives and all the positives. In this case, it would be the measure between the correct number of patients with malaria out of all the patients that have it. When the cost of false positives is high, precision is helpful.

Precision and recall are the foundations of this F1-Score. This is the harmonic mean of the combination of both measures relative to a specific positive class. One being its best, and 0 its worse. An excellent F1 score means we have a low number of false positives and low false negatives. This measure is preferred for imbalanced datasets over the previously discussed ones.

Area Under Curve (AUC) is usually used in binary classification problems. It measures the entire two-dimensional area under the ROC curve. The ROC curve displays the relationship between the false-positive and true-positive rates for different probability thresholds of model predictions.

The confusion matrix is the easiest way to display true positives, true negatives, false positives, and false negatives. True positives correspond to the number of patients correctly classified as parasitized; true negatives correspond to those who do not have the disease. False negatives are the main case we want to avoid in medical practice as they represent those with the disease, but the algorithm misclassified them as healthy while they are sick. They are also known as type I error. False positives are also a case to pay attention to as they are patients who do not have malaria, but the model is labeled as having such. This is known as a type II error.

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

a)

$$\text{Recall} = \frac{TP}{TP + FN}$$

b)

$$\text{Precision} = \frac{TP}{TP + FP}$$

c)

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

d)

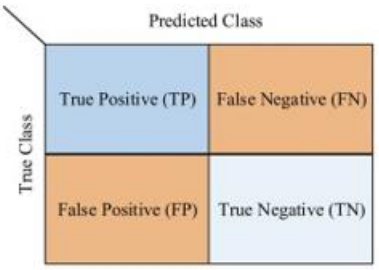

Figure 10 - The formulas for each of the metrics: a) accuracy, b) recall, c) Precision, d) F1-Score, and e) confusion matrix.

5. RESULTS AND DISCUSSION

For the scope of this work, it was hypothesized that lightweight pretrained models have the potential to reach outstanding results despite their lower number of parameters. The chosen metrics were binary accuracy, recall, precision, F1-Score, and AUC score to evaluate their performance. The confusion matrix was also an essential visual method to understand where the models were misclassifying classes of false negatives and false positives, both problematic in the medical field as the first can lead to preventable deaths and the latter to a poorly administration of drugs which contributes to the already problematic drug resistance.

Two methods of transfer learning were applied. For the first experiment, the last layer was removed and replaced by a dense layer with one neuron and a sigmoid activation function. Before that, only the global average pooling and dropout rate of 0.5 was used for regularization purposes. Several tests were done to determine the ideal amount to use in dropout, with the values 20%, 50%, and 70%. The first value led to a longer training time with an average of 17 epochs across all models. The last one, 70%, lead to slight overfitting with 13 epochs across all metrics. The value of 50% led to an average of 19 epochs. Although training took a long time, significant overfitting cases were prevented compared to the previous regularization values. For this very reason, it was chosen as the standard for the scope of this work. The entire display of its results can be seen in table 7.

Table 7 - Experiment 1 Results

Model	Accuracy	Loss	Precision	Recall	F1-Score	AUC	Time (s)	Epochs
MobileNet	0.995	0.016	0.995	0.995	0.995	0.995	1925	16
MobileNetV2	0.994	0.020	0.994	0.994	0.994	0.994	1694	22
NASNetMobile	0.994	0.034	0.994	0.994	0.994	0.994	8090	30
EfficientNetB0	0.996	0.017	0.996	0.996	0.996	0.996	805	9

The overall best model was EfficientNetB0, with a score of 99,6% across all metrics and converging in only nine epochs out of 30. Of all models, it was the one with the minor false negatives and positives, with 6 and 7, respectively, out of support of 1314 for each class. The second best was MobileNet, with a loss of .0016 and following metrics of 99.5%, a training time of 1925 seconds, and 16 epochs until convergence. MobileNetV2 and NASNetMobile came last with identical scores. It is worth noting that MobileNetV2 started to overfit and most likely kept its good scores thanks to the combination of the Keras callbacks. When comparing the number of false negatives and positives, there is no place for much disparity among them since the numbers are low, to begin with. MobileNet displays 13 misclassified labels, MobileNetV2 a total of 15, NASNetMobile follows with 14, and EfficientNetB0 with only 10. The detail of misclassification across classes can be seen in figure 11 below.

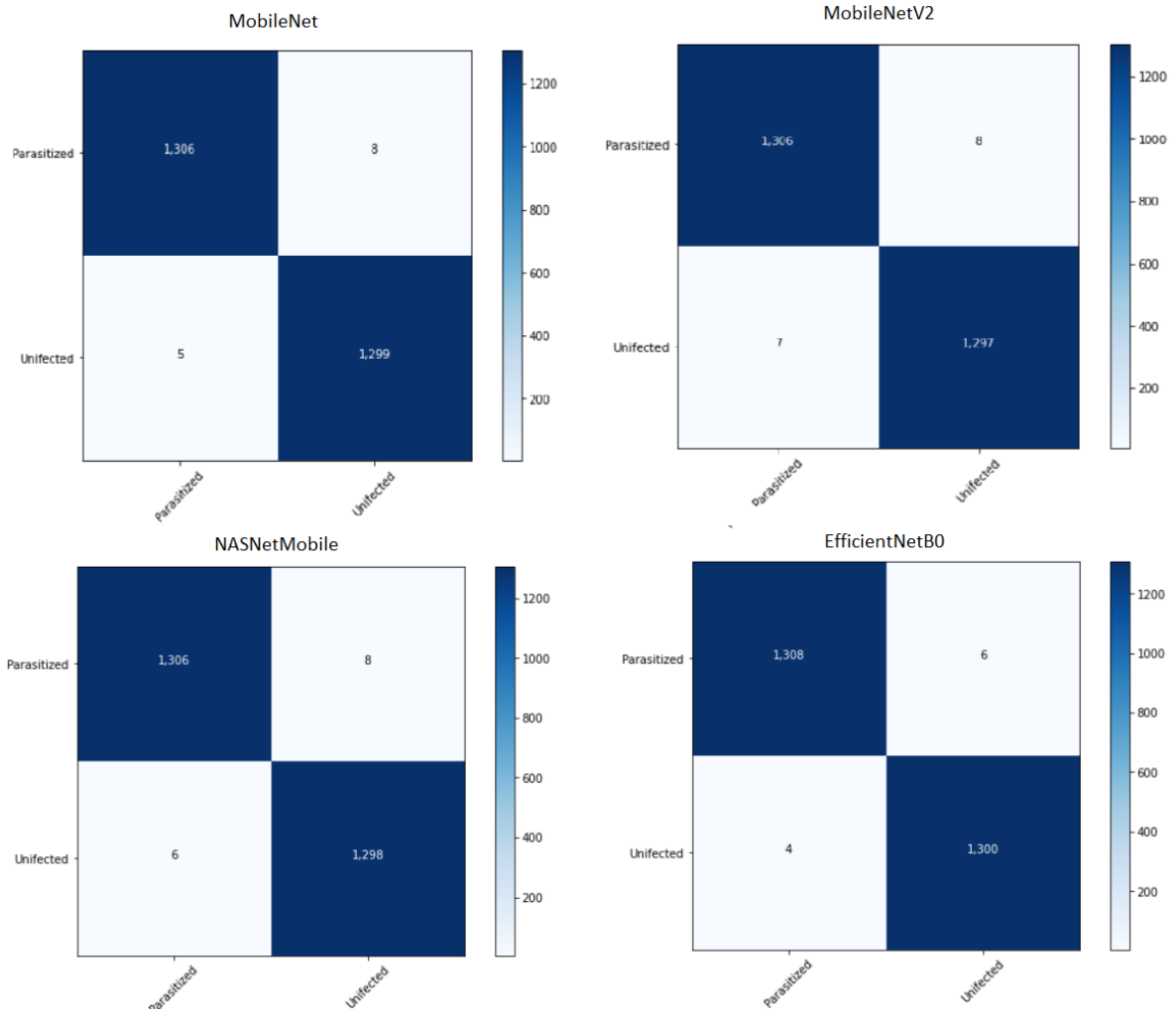

Figure 11- Confusion Matrix of experiment number one

For the second experiment, the last layers were frozen. Since there is no way to know the correct number of ideal layers to freeze, both the last three and, then the last four layers were frozen on all models. Overall, there was a loss in accuracy, recall, precision, and F1-Score of around 1% when both experiments were considered, with the first being the most successful. Although some models display signs of overfitting, the model checkpoint saved the best version of each, which may explain each model's outstanding performance even if overfitting happens.

This second experiment is subtle if the objective is to warn that studying the most critical layers must be done carefully. The process of freezing layers and testing their importance can also be time-consuming. The consistency between freezing the last three and the last four layers is the performance of EfficientNetB0, where it remains more potent than its counterparts regarding scores and the least low number of misclassified images.

Table 8 - Experiment 2: Results with three frozen layers

Model	Accuracy	Loss	Precision	Recall	F1-Score	AUC	Time (s)	Epochs
MobileNet	0.977	0.079	0.977	0.977	0.977	0.977	446	12
MobileNetV2	0.984	0.058	0.984	0.984	0.984	0.984	464	12
NASNetMobile	0.985	0.060	0.985	0.985	0.985	0.985	1014	15
EfficientNetB0	0.987	0.048	0.987	0.987	0.987	0.987	523	14

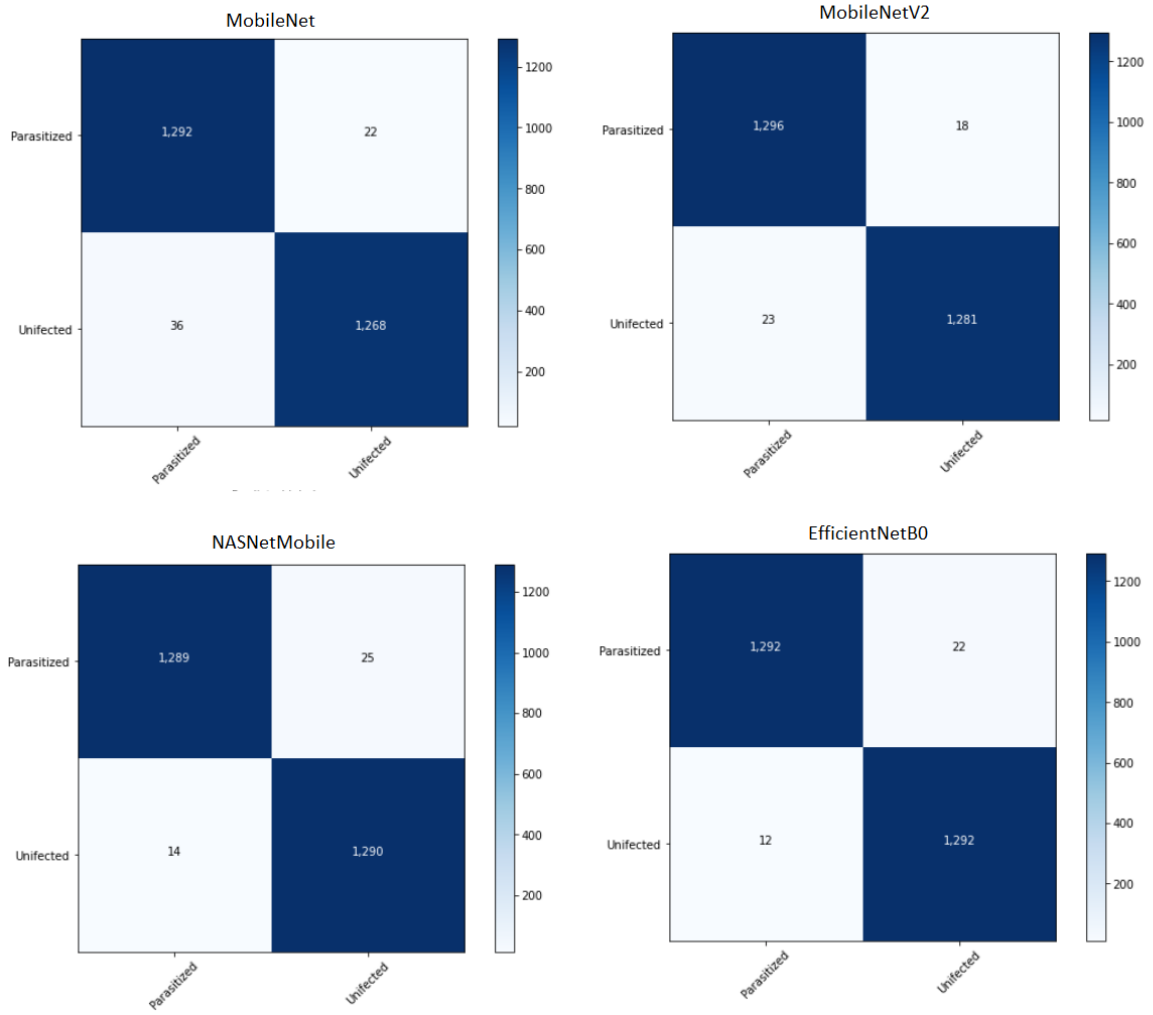

Figure 12 - Confusion Matrix of experiment number two with three frozen layers

Table 9 - Experiment 2: Results with four frozen layers

Model	Accuracy	Loss	Precision	Recall	F1-Score	AUC	Time (s)	Epochs
MobileNet	0.983	0.052	0.983	0.983	0.983	0.983	405	11
MobileNetV2	0.978	0.077	0.978	0.978	0.978	0.978	397	11
NASNetMobile	0.980	0.053	0.980	0.980	0.980	0.980	1119	15
EfficientNetB0	0.984	0.049	0.984	0.984	0.984	0.984	540	13

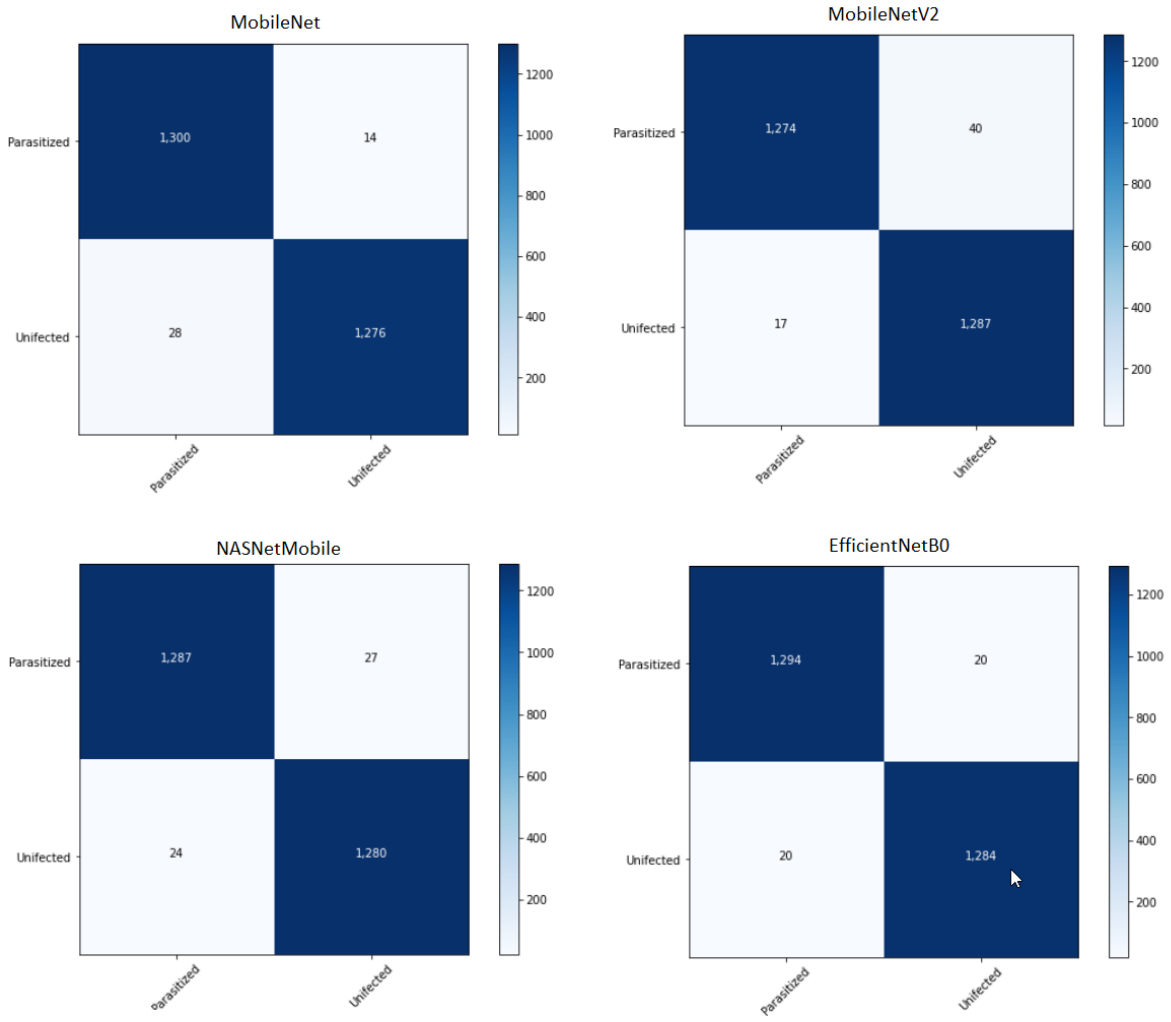

Figure 13 -Confusion Matrix of experiment number two with four frozen layers

It is worth noting that the stated results discussed so far are the ones with no data augmentation. Contrary to what most research states about the improvement of results with the use of data augmentation, in our case, it did not lead to such an improvement and severely increased the time of training and number of epochs, often to reach the same or very similar result as with no data augmentation. It goes according to the example of section 4.2.1. but not for the same reason that extreme image distortion was not the case. It might be the case that the new generation examples are not contributing to the mitigation of the only problem left in the dataset after correcting the wrongly labeled example: the difference between colors and the possibility of staining artifacts. These issues would have to be tackled by lighting techniques to make images more similar among themselves color-wise. The new examples do not add significant further information, and their generation, even with the image generator class, is weighing on training time.

Overall, it was shown that pretrained lightweight models can do very well in malaria classification and that more parameters are unnecessary to achieve the same result. It would be interesting to compare these results with the ones mentioned in section 3.3.1 regarding works that also used transfer learning. Some of them might have used the original dataset of this work and not the curated one done here, so it would be somewhat dubious about making a fair comparison since this dataset, despite its slight imbalance, has fewer false negatives and false positives. Most of the pre-trained models used in said studies achieved excellent results and are much larger. The models used in this work can also be deployed when used with the first approach as a standard due to their performance.

The chosen model for pruning was MobileNet, as implemented in experiment number one. Of all models, it did not show signs of overfitting throughout the epochs and achieved a very satisfying result. While true that it was EfficienetNetB0 that performed the best performance in the least amount of time, unfortunately, it is not yet possible to prune it as a pretrained model via the Keras API. The second best option is MobileNet due to its score and lack of overfitting. When trained with a sparsity of 50%, the scores presented across all metrics were 99.4% during 15 epochs. When sparsity was increased to 70% and 90%, these increased to 99.5% while their sizes dropped to 7.2666 MB, 5.0512 MB, and 4.1926 MB for 50%, 70%, and 90% sparsity, respectively. Interestingly, we can prune a model in the last percentages and not lose accuracy, with particular attention to the last case.

Table 10 - Pruned MobileNet with different sparsity sizes. Pruned with polynomial decay

Sparsity	Accuracy	Loss	Precision	Recall	F1-Score	AUC	Time (s)	Epochs	Size (MB)
50	0.994	0.015	0.994	0.994	0.994	0.994	2092	15	7.2666
70	0.995	0.015	0.995	0.995	0.995	0.995	2511	19	5.0512
90	0.995	0.014	0.995	0.995	0.995	0.995	2202	18	4.1926

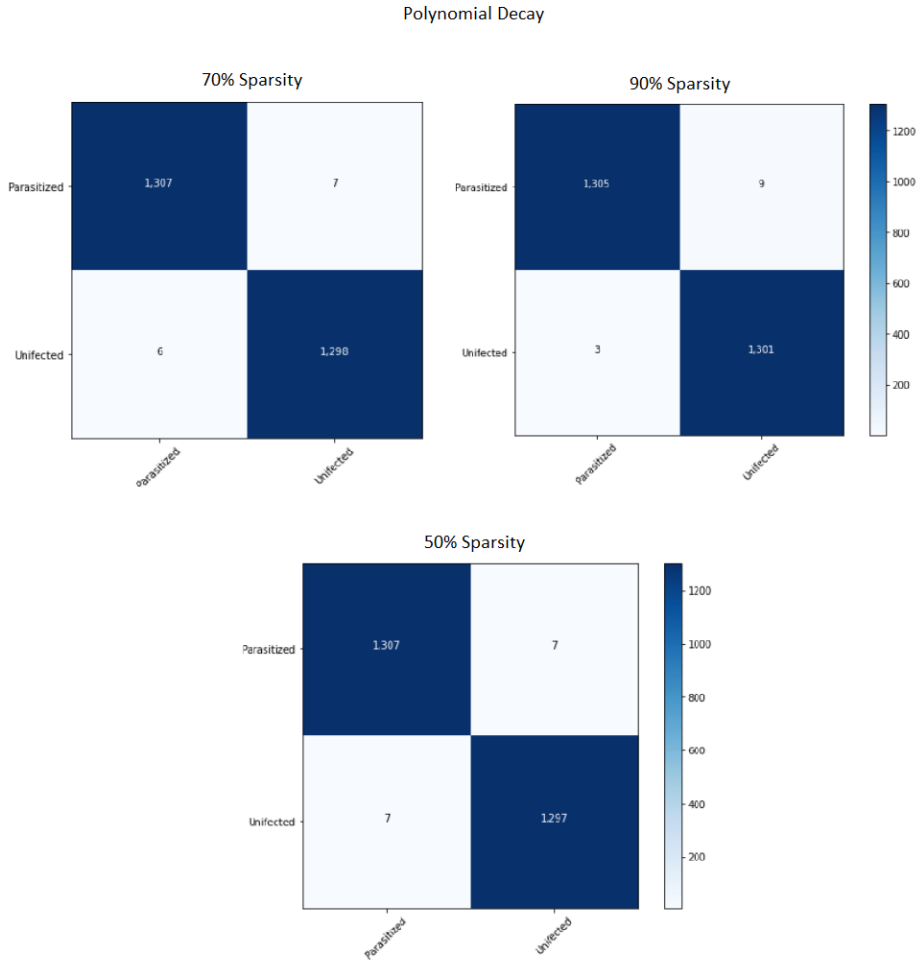

Figure 14 - Confusion Matrix of pruning with polynomial decay

Table 11- Pruned MobileNet with different sparsity. Pruned with constant sparsity

Sparsity	Accuracy	Loss	Precision	Recall	F1-Score	AUC	Time (s)	Epochs	Size (MB)
50	0.993	0.0196	0.993	0.993	0.993	0.993	2195	16	7.2666
70	0.995	0.0206	0.995	0.995	0.995	0.995	1759	14	5.0512
90	0.995	0.0193	0.995	0.995	0.995	0.995	2520	20	2.6631

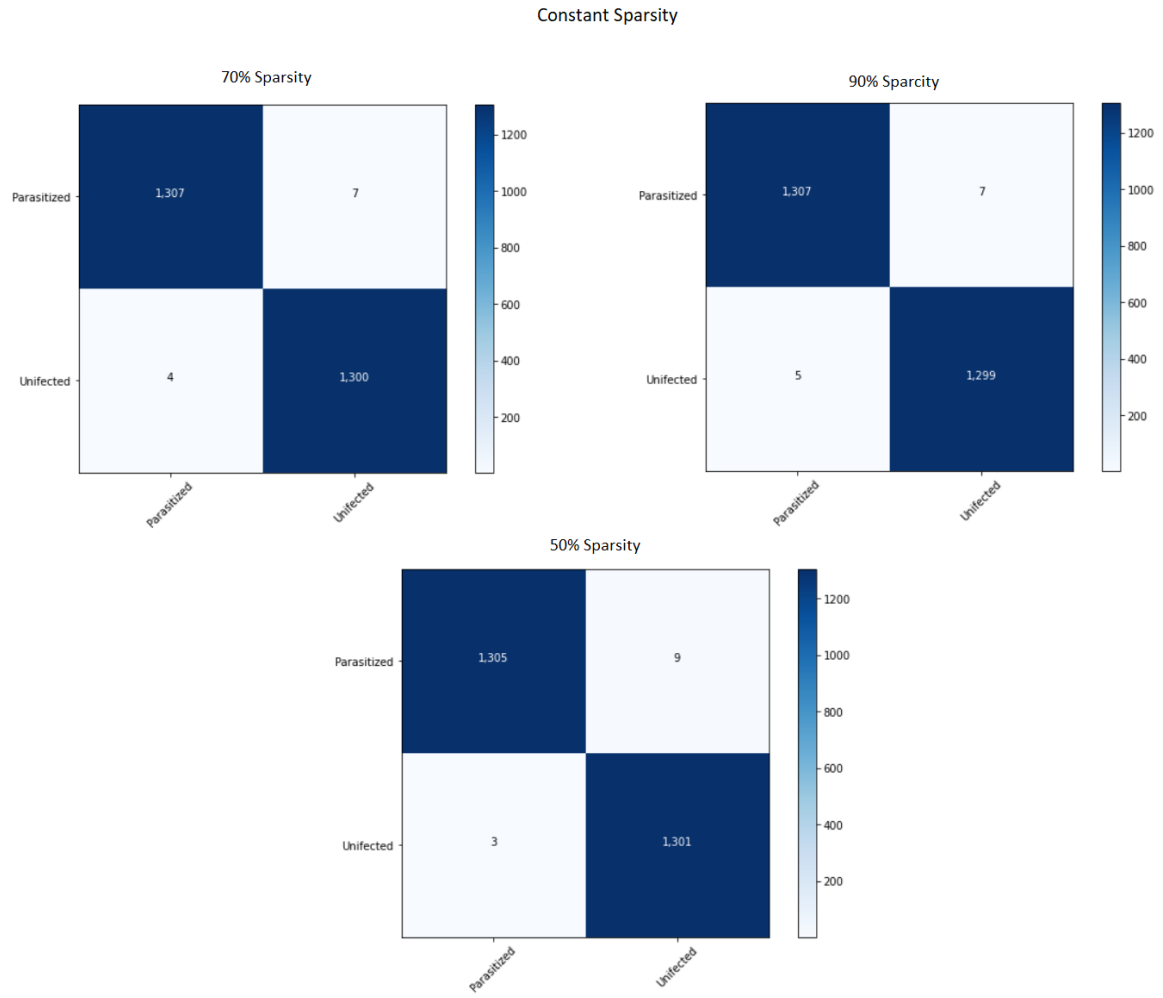

Figure 15 - Confusion Matrix of pruning with constant sparsity

Regarding accuracy, precision, recall, and f1-score, MobileNet pruned via constant sparsity keeps its 99,5% when pruned for 90% of its weights and drops to a size of 2.669 MB.

Although there was no significant difference between the two types of pruning regarding results, it was indeed different regarding the models' final size. Constant sparsity showed the highest degree of compression with 70% sparsity. Surprisingly, it was at 50% sparsity that the accuracy dropped on this section. One might expect that the more aggressive the compression, the more the model would suffer and perform. However, this was not the case. With polynomial decay, the difference in ending sizes was more gradual and not as steep, and the accuracy was not lost. Given all this, if one final model had to be chosen, it would be MobileNet pruned with 70% constant sparsity for its' scores of 99.57% across accuracy, recall, precision, and f1-Score while going from the original MobileNet size of 16 MB to 2.6631 MB.

6. CONCLUSIONS

Due to the dire malaria situation, there is an urgent need for methods of diagnosis and treatment capable of reaching all endemic zones where the vector mosquito is endemic. There is a significant disparity in access to healthcare between cities and rural and more remote places. The microscopy-based diagnosis has been dubbed the golden standard by WHO due to its versatility and reliability. However, it has a few shortcomings: it demands a high number of personnel with training and knowledge to interpret the results, and its portability is compromised due to the lack of electricity in some remote places. Thanks to the advance of technology, it is now possible to attach a cheap mobile phone to a microscope, making it portable and functional during long hours without recharge. This, combined with the automation of malaria detection, also diminishes the need for highly trained professionals in terms of numbers while providing a reliable diagnosis.

Extensive work has been done in malaria detection, including using CNNs. However, many of these works focus on the performance alone, leaving behind the size and possibility of being deployed. Large models are rarely compatible with devices with lower computing power, like mobile phones. This work's central premise is that more parameters are not necessarily needed to achieve peak performance. To measure the performance, the metric used were binary accuracy, precision, recall, F1-Score, and AUC score, while making use of the confusion matrix to understand better where the models were going wrong in terms of false positives ad false negatives, as these are crucial concepts to avoid in any medical field. The chosen pre-trained models were MobileNet, MobileNetV2, EfficientNetB0, and NASNetMobile; all models were conceived to work on devices like mobile phones, deeming them suitable for our task. Because models are also highly dependent on the choice of the best possible hyperparameters, a starting learning rate of 0.001 and a Nadam optimizer were chosen. They were seen to work very well in conjunction. Keras callbacks, such as early stopping and model checkpoint, played a role in stopping overfitting by diminishing the learning rate as validation loss stopped improving and by saving the best possible score of the model even when the performance decreased as more epochs passed without improvement. As a regularization technique, dropout with a value of 50% was seen to work the best. Two transfer learning techniques were employed. All models were trained with the ImageNet dataset. So, the last layer was replaced with a dense layer with the sigmoid function. Before that, global average pooling and dropout were used. As for the second experience, the last layers were frozen and dense, and dropout layers with different values were added before the previous dense layer. The final experiment has decreased across all metrics by about 1%. Across all experiments, EfficientNetB0 was the best overall model, reaching 99.6% in all scores. Unfortunately, it was impossible to show if such performance was kept via pruning, as the Keras optimization API does not yet support it. The second-best model was MobileNet, as tested in the first experiment, with scores reaching 99.5%.

Not every weight contributes the same for CNN, even with some not contributing much, even if at all. These less valuable weights were discarded via the pruning technique. Various degrees of pruning was tested to show how much one model can be pruned without the loss of accuracy, recall, precision, AUC, and F1 score. Among them, sparsity was analyzed with 50%, 70%, and 90% with both constant sparsity and polynomial-based decay. MobileNet was compressed following all these requirements. It was observed that we could prune this particular model for this dataset with the same structure for

experiment number one with the sparsity of 70% with constant sparsity that not only is the performance not lost, but the size is decreased up to 2.6631 MB.

Data augmentation did not show any improvement in any model, despite the amount of research showing its benefits in image classification. This is not to disregard the vast number of studies confirming its importance in deep learning. However, the geometrical transformations employed in this work were proven not helpful for this particular kind of data. Although the dataset is curated, differences in lighting acquired during the blood staining itself and other artifacts might imply that maybe data augmentation with a focus on color distribution may be proven to be most adequate for this case.

7. LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORKS

To end this work, some final considerations must be addressed. Besides these excellent results acquired for the models even with compression, further investigation would be required for deployment in the real world:

- It is worth noting that, despite the dataset not being fully balanced, the difference between classes was insignificant, and corrections to originally misclassified labels were made. This is often not the case in medical imaging, where data tends to suffer from imbalance. When collecting blood samples, healthy red blood cells will always be much more prevalent than parasitized cells.
- The only training was between cells infected with one parasite in the ring stage, while there are four main parasites, all with a complex lifecycle where its multistage differ in shape and size. To take full advantage of the microscopy capabilities, an ideal algorithm should be trained on all of these, as they have a role in determining the stage and progression of malaria.
- Other hyperparameters can also be tested for further improvement. Does Adam not show any of its shortcomings in this example? Is there another learning rate that approximates closer to its optimal?
- For this work, the model was pruned. A search is conducted across the entire network, looking for the weights that have the most negligible impact, but there is an option only to prune the desired layers. We can choose which layers are based on their type or name. It would be interesting to see if, by pruning only a certain number of layers, the model size could be minimized while surpassing the initial accuracy and other desired metrics.

8. BIBLIOGRAPHY

A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: efficient convolutional neural networks for mobile vision applications (2017). arXiv:1704.04861

Arshad, Qazi Ammar, et al. A Dataset and Benchmark for Malaria Life-Cycle Classification in Thin Blood Smear Images. Neural Computing and Applications, vol. 34, n. 6, 2022, pp. 4473–85. DOI.org (Crossref), https://doi.org/10.1007/s00521-021-06602-6.

Bibin, D., Nair, M. S., & Punitha, P. (2017). Malaria Parasite Detection From Peripheral Blood Smear Images Using Deep Belief Networks. IEEE Access, pp. 5, 9099–9108. doi:10.1109/access.2017.2705642

Chen, J., (2022). An updated overview of recent gradient descent algorithms. [online] Johnchenresearch.github.io. Available at: <https://johnchenresearch.github.io/demon/> [Accessed 2 October 2022].

Dong Y., Jiang Z., Shen H., Pan W.D., Williams L.A., Reddy V.V., Benjamin W.H., Bryan A.W. Evaluations of deep convolutional neural networks for automatic identification of malaria-infected cells; Proceedings of the 2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI); Orlando, FL, USA. 16–19 February 2017; pp. 101–104

Elgendi M, Nasir MU, Tang Q, Smith D, Grenier JP, Batte C, Spieler B, Leslie WD, Menon C, Fletcher RR, Howard N, Ward R, Parker W, Nicolaou S. The Effectiveness of Image Augmentation in Deep Learning Networks for Detecting COVID-19: A Geometric Transformation Perspective. Front Med (Lausanne). 2021 Mar 1;8:629134. doi: 10.3389/fmed.2021.629134. PMID: 33732718; PMCID: PMC7956964.

Eze PU, Asogwa CO. Deep Machine Learning Model Trade-Offs for Malaria Elimination in Resource-Constrained Locations. Bioengineering (Basel). 2021 Oct 21;8(11):150. DOI: 10.3390/bioengineering8110150. PMID: 34821716; PMCID: PMC8614791.

Fuhad, K. M. F., Tuba, J. F., Sarker, M. R. A., Momen, S., Mohammed, N., & Rahman, T. (2020). Deep Learning Based Automatic Malaria Parasite Detection from Blood Smear and Its Smartphone Based Application. Diagnostics, 10(5), 329. doi:10.3390/diagnostics10050329

Gopakumar G.P., Swetha M., Sai Siva G., SaiSubrahmanyam G.R.K. Convolutional neural network-based malaria diagnosis from focus stack of blood smear images acquired using custom-built slide scanner. J. Biophotonics. 2018;11:e201700003. doi: 10.1002/jbio.201700003.

Islam MR, Nahiduzzaman M, Goni MOF, Sayeed A, Anower MS, Ahsan M, Haider J. Explainable Transformer-Based Deep Learning Model for the Detection of Malaria Parasites from Blood Cell Images. Sensors. 2022; 22(12):4358. https://doi.org/10.3390/s22124358

Janiesch, C., Zschech, P. & Heinrich, K. Machine learning and deep learning. Electron Markets 31, 685–695 (2021). https://doi.org/10.1007/s12525-021-00475-2

Li, Sen & Du, Zeyu & Meng, Xiangjie & Zhang, Yang. (2021). Multi-stage malaria parasite recognition by deep learning. GigaScience. 10. 10.1093/gigascience/giab040.\

Linder N, Turkki R, Walliander M, Mårtensson A, Diwan V, Rahtu E, Pietikäinen M, Lundin M, Lundin J. A malaria diagnostic tool based on computer vision screening and visualization of Plasmodium falciparum candidate areas in digitized blood smears. PLoS One. 2014 Aug 21;9(8):e104855. DOI: 10.1371/journal.pone.0104855. PMID: 25144549; PMCID: PMC4140733.

Malhotra, R., Joshi, D., & Shin, K.Y. (2020). Approaching Bio Cellular Classification for Malaria Infected Cells Using Machine Learning and then Deep Learning to compare & analyze K-Nearest Neighbours and Deep CNNs. ArXiv, abs/2005.11417.

Militante, Sammy V., e Renante A. Diamante. Malaria Disease Diagnosis from a Blood Smear Samples using the Deep Learning MobileNet Models. 2021 Fourth International Conference on Vocational Education and Electrical Engineering (ICVEE), 2021, pp. 1–6. IEEE Xplore, https://doi.org/10.1109/ICVEE54186.2021.9649688.

Mohanty, I., Pattanaik, P. A., & Swarnkar, T. (2019). Automatic Detection of Malaria Parasites Using Unsupervised Techniques. Lecture Notes in Computational Vision and Biomechanics, 41–49. doi:10.1007/978-3-030-00665-5_5

Montalbo, F., Alvin, A., Empirical Analysis of a Fine-Tuned Deep Convolutional Model in Classifying and Detecting Malaria Parasites from Blood Smears. KSII Transactions on Internet and Information Systems, vol. 15, n. 1, 2021. DOI.org (Crossref), https://doi.org/10.3837/tiis.2021.01.009.

Pirnstill, C. W., & Coté, G. L. (2015). Malaria Diagnosis Using a Mobile Phone Polarized Microscope. Scientific Reports, 5(1). doi:10.1038/srep13368

Poostchi, M., Silamut, K., Maude, R. J., Jaeger, S., & Thoma, G. (2018). Image analysis and machine learning for detecting malaria. Translational Research, 194, 36–55. doi:10.1016/j.trsl.2017.12.004

Rahman, Aimon, et al. Improving Malaria Parasite Detection from Red Blood Cell using Deep Convolutional Neural Networks. arXiv, 2019. arXiv.org, https://doi.org/10.48550/arXiv.1907.10418

Rajaraman S., Jaeger S., Antani S.K. Performance evaluation of deep neural ensembles toward malaria parasite detection in thin-blood smear images. PeerJ. 2019

Rajaraman, S.; Antani, S.K.; Poostchi, M.; Silamut, K.; Hossain, A.; Maude, R.J.; Jaeger, S.; Thoma, G.R. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. PeerJ 2018, 6, e4568.

Reynolds, A., (2022). Anh H. Reynolds. [online] Anh H. Reynolds. Available at: https://anhreynolds.com/blogs/cnn.html [Accessed 8 October 2022].

Rosado L., Da Costa J.M.C., Elias D., Cardoso J.S. Automated detection of malaria parasites on thick blood smears via mobile devices. Procedia Comput. Sci. 2016; 90:138–144. doi: 10.1016/j.procs.2016.07.024

Sriporn, K.; Tsai, C.-F.; Tsai, C.-E.; Wang, P. Analyzing Malaria Disease Using Effective Deep Learning Approach. Diagnostics 2020, 10, 744

Tan, M. and Le, Q.V. (2019) EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, 9-15 June 2019, 6105-6114

Z. Liang, A. Powell, I. Ersoy, et al., “CNN-based image analysis for malaria diagnosis,” in Proceedings of the 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 493–496, Shenzhen, China, December 2016.

Zhu H., Mavandadi S., Coskun A. F., Yaglidere O. & Ozcan A. Optofluidic fluorescent imaging cytometry on a cell phone. Anal Chem 83, 6641–6647 (2011)