Given NIH blood smear dataset use pattern recognition to detect automatically malaria parasites
Final Project Report

Margaux Törnqvist
ENS Paris-Saclay
margaux.tornqvist@ens-paris-saclay.fr

Abstract

Malaria is an infectious disease which causes hundred and thousands of deaths each year. The diagnosis of malaria parasite is very time consuming and burdensome for trained clinicians: they need to count manually the number of infected blood cells with a microscope. An efficient deep learning solution able to recognize automatically Plasmodium in microscopic blood smear images could reduce clinical burden and time for doctors and trained clinicians. If deep learning based techniques seem to be very efficient in image classification tasks, they are often computationally expensive and consequently energy and memory consuming. In poverty-stricken areas, which are the most affected by malaria, it would be preferable to have a solution easy to deploy offline on a low-coast smart-phone. In order to take in account this constraint, this work will compare performance and size of existing deep learning networks. Moreover, this work will try to leverage the size of deep networks and compare traditional networks to a basic solution which encodes features into a latent space : an auto-encoder. This work will focus on evaluating this simple auto-encoder architecture which is way smaller than deep networks and which may be a compatible solution to our problem.

1. Introduction

1.1. Motivations

Malaria is a life-threatening infectious disease caused by Plasmodium parasites spread through mosquitoes' bites. After that the parasites mature in the liver, they infect blood cells. WHO estimates that in 2018 there was 228 billion cases worldwide, and more than 400 000 deaths [1]. Africa region was home of 93 percent of them and 94 percent of deaths. Moreover, there is a high correlation between poverty and malaria cases. Indeed, poor regions do rarely have access to proper healthcare. And without good healthcare and treatment, the disease outbreak can be difficult to contain. Early diagnosis can prevent the disease from spreading and avoid deaths. The most frequent diagnosis method is using blood smears. A drop of blood of the patient is spread on a glass slide to make a "blood smear", which is stained with an agent to highlight malaria parasites. Further a clinician examines the smear with a microscope and manually counts how many parasites are present in approximately 5000 blood cells. This process is extremely fastidious, time-consuming and prone to error. A deep learning solution to automatically detect malaria parasites from blood smears with a high accuracy would save the precious time of clinicians and doctors.

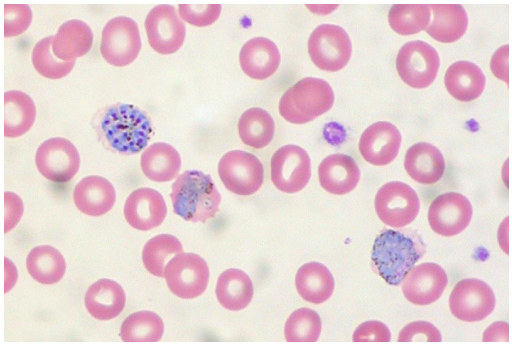

Figure 1. Blood smear microscopic image

If deep learning based techniques seem to be very efficient in image classification tasks, they are often computationally very expensive. For instance, a model such as ResNet-50 has a size of approximately 200 MB and requires 3.8G FLOPs computations. On a representative low coast smartphone's CPU, for instance Qualcomm Snapdragon 625, inference time for such a model is about 616 ms, which definetely impacts the user's experience and drains the battery [2]. In order to use an automatic malaria classifier on a mobile application, the smartphone needs a charged battery, power, and preferably be connected to the cloud and

therefore an internet connection. But in poverty-stricken areas, the combination of all these elements may be difficult. Therefore, it would be desirable to have a smaller model with a high accuracy and which can be easily deployed offline on a low-coast smartphone. Moreover, training a small model is also less energy and time consuming which allows to bring a solution fast in the field and therefore help quickly doctors and clinicians during an outbreak.

2. Problem definition

In this work, we will tackle a real-world problem and be careful to take in account the constraints imposed by the context of use of the solution. How can we develop a efficient model to detect malaria prediction in blood smears ? How can we make the solution usable in poverty-stricken areas i.e computationally efficient in low resource settings ? How can we leverage our model in order to reduce it's size ?

The project's objectives are the following:

1. Create a deep learning model that can match the performance of the NIH i.e a test accuracy above 95 , which is already remarkable.

2. Create a model with considerably smaller size in order to be deployed on low-coast smartphones.

We will train multiple accurate and computationally efficient models for malaria parasite detection in blood smear images from NIH's publicly available malaria dataset. First we will train models such as residual networks to reach NIH's performance. We will adapt our training depending on our computational resources which are considerably less important than NIH's. Then we will try to leverage the size of our model in order to reach viable computational resources. We will train smaller models such as MobileNet and SqueezeNet which are known to be compatible with deployment on mobile devices. Finally, we will try to implement the latest work produced by Fyhad et al [7]: an auto-encoder network. It's architecture is particulary interesting for us by it's relatively small number of convolutional layers. Indeed, by compressing information into a latent space of smaller dimensions than the original problem, it reduces the number of parameters to learn for classifying features.

For each classifier we seek to minimize a classification error. We mesure this error between the probability distribution of the two classes of interest by using cross-entropy. As we are in a binary classification problem, we can predict the input of being in the positive class "1" with a probability p, and in the negative class "0" with probability 1-p. Finally, our problem is framed as minimizing the cross-entropy which is the average cross-entropy over both classes:

Objective function:

$$L_\theta = -(ylog(p) + (1-y)log(1-p))$$

3. Related work

As malaria is a life threatening disease, many researchers have worked on the automatic pattern recognition of malaria in blood smear images. First, researchers were interested in automatic systems relying on machine learning techniques. For instance, [3] tried to apply Support Vector Machines(SVM) and Principal Component Analysis(PCA) on extracted morphological features from the images. Such techniques suggest hand-engineered features for decision-making and demands expertise in analysis of the region of interest in microscope images. Moreover, accuracy achieved by these techniques is low compared to recent research using deep learning.

With the boom of deep learning, several teams have tried to automatically detect malaria in microscope blood smear images using CNNs. Most of the studies try to improve accuracy using different traditional or customized CNNs. Dong et al., trained three different models : LeNet5, ALexNet and GoogLeNet. They also trained an SVM classifier and concluded that CNNs definitely outperform classic ML techniques [4]. In 2016, Liang et al., reported 97.4% accuracy with their custom 16 layer model CNN and exceeded performances obtained with transfer learning [5]. Moreover, in 2018 Rajaraman et al. from NIH [6], compared performances of AlexNet, VGG-16, Xception, DenseNet-121, ResNet-50 on a dataset they made publicly available. Best performance, 95.7% was obtained by fine tuning a pretrained ResNet50. Finally, during the same year, Fuhad et al., a team based in Bangladesh, revisited the NIH's work in order to bring a viable solution with a smartphone's computational capacity. In their latest paper [7], they propose an auto-encoder architecture which performs with an accuracy of 99.23% with only 4600 FLOPs. Their model sizes ruffly 70 kB and is compatible with low resource settings. Indeed they have tested their solution on multiple mobile devices in both offline and online mode and report that they are able today to predict an accurate classification of malaria in blood smear images within 1s.

4. Methodology and materials

In order to conduct our different experiments, we will use the publicly available malaria dataset provided by NIH [5]. In the following subsections is detailed how data was collected and preprocessed. In the architecture subsections, we discuss how we chose our best model in terms of best performances, ability to generalize and computational effectiveness. We also describe how we tried to leverage the size of our model in order to make it as small as possible. Experimental settings are described in training details subsections.

4.1. Data Collection and preprocessing

The dataset included 27,558 (350 MB) cell images with equal instances of parasitized and uninfected ones. A sample which is parasitized means that there is presence of Plasmodium, and uninfected that there is no Plasmodium. The data was taken from 150 different Plasmodium falciparum and 50 healthy patients. The parasites were stained with the agent Giemsa. The pictures were taken at Chittagong Medical College Hospital in Bangladesh using a smartphone by placing it on a conventional microscope. Therefore, blood cells were segmented out of these pictures and the final database was then labelled by researchers at the Mahidol-Oxford Tropical Medicine Research Unit in Thailand. Somehow article [7] seem to claim that approximately 1 300 images may be mislabeled and propose a correction of the labelling in a publicly accessible google drive [8]. We chose to remove them by precaution from our data set. In Table 1 is depicted the training/validation/test split which was done following the rule 0.9/0.1 for training and testing, 0.8/0.2 for training and validation. The removal of the 1 300 images explains the slightly imbalanced data set.

Dataset	Parasitized	Uninfected	Total
Training	9389	9496	18885
Validation	1067	1031	2098
Test	2709	2537	5266

Table 1. Dataset table

In deep learning, data preparation plays a fundamental role in model performance and ability to generalize. The images were re-sampled to 224 × 224 and 64 x 64. pixel resolutions to suit the input requirements of the pretrained CNNs and normalized in order to boost convergence. The data augmentation applied is described in Table 2. It helps our model to prevent from overfitting and improves performance.

Augmentation type	Parameters
Random Rotation	20
Random Scale	[0,0.05]
Width/Height Shift	[-0.05,0.05]
Shear Intensity	[-0.05,0.05]
Horizontal Flip	True

Table 2. Data Augmentation Table

4.2. Transfer learning with pretrained CNNs

First, we chose to train traditional CNNs efficient in computer vision tasks and reach NIH's performances described in [].We evaluated the performance of pretrained CNNs such as ResNet-50, MobileNetV2 and SqueezeNet

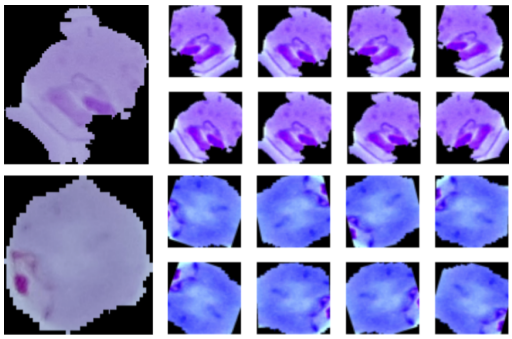

Figure 2. Data augmentation applied on input images

towards extracting the features from the parasitized and uninfected cells.

In [6], they explain how they obtained their best model with an accuracy of 0.957 on test set by using transfer learning with ResNet-50 pretrained on ImageNet. In short, ResNet-50 is a residual network created by Microsoft research team in 2015, which's architecture has the particularity of integrating some skip connections between lower and higher level layers. These shortcuts between layers, ensure coarse information is communicated along the network and that deep layers are kind of "referenced to layer inputs". This ensures deep layers perform as well as lower layers. Moreover, Resnets are easier to optimize than classical deep networks. Indeed, their architecture help not falling into the problem of vanishing gradients which occurs typically in very deep layers. Consequently, this allows us to build deeper networks which improve performance in several classification tasks. With the same training settings, we fine-tuned a pretrained ResNet-50 on ImageNet to learn how to extract features. We optimized a binary crossentropy loss with SGD, a learning rate of 1e-5, momentum of 0.9 and l2-regularizer of 1e-6. As the number of initializations weren't specified in the paper we chose to train the model over 100 epochs. The results weren't the one promised by the paper so we chose to change our training settings in order to reach NIH's performance. Indeed, NIH team seems to have trained the model over several hundred of epochs to reach their accuracy above 0.99. Using a GPU of Google Colab, we have less computational resources. Thus we chose to start training from a coarser learning rate and decrease it gradually. Training was done over 50 epochs with SGD optimizer initialized using a multiplicative learning rate of 1e-2 with gamma = 0.9478, step = 1, and a momentum of 0.9. This conducted to similar results, an accuracy of 0.992, as described in the paper.

Furthermore, we decided to explore how we could reduce the size of this efficient network. We chose to simply train smaller ResNets such as ResNet-34 and ResNet-18 which have similar architectures than ResNet-50 but less filters in the convolutional layer blocks. Accuracies reached

respectively 0.98 and 0.9, which is slightly lower than with ResNet-50. We then decided to propose two new architectures to solve the problem : MobileNetV2 and SqueezeNets. Why did we chose to study these two architecture in particular ? Like MobileNetV2 name's suggests, this network is relatively small and was designed to be deployed on mobile phones. MobileNetV2 uses as its first version, MobileNetV1, proposed by Google in 2017, depthwise convolutional layers. This new layer design optimizes the classical convolution making its execution faster,less memory intensive and thus less energy consuming. MobileNetv2 built upon its first version adds two features to the architecture : linear bottlenecks between the layers and residual shortcuts between the bottlenecks. We chose SqueezeNet for it's AlexNet-level performance achieved with 50 times less parameters, 3 times less inference time and sizes 500 times less with a model size inferior to 0.5 MB. SqueezeNet's architecture tries to integrate three main strategies in order to build a compressed and efficient network. By replacing 3x3 convolutions by 1x1 convolutions and decreasing the number of input channels to 3x3 they manage to leverage the number of parameters by maintaining accuracy. By downsampling as late as possible in the network in order to preserve a maximum number of large activation maps, which they believe improves performance, they attempt to increase accuracy with as few as possible parameters.

After judiciously looking for the best model architecture to decrease the size of our classifier, we trained a MobileNetV2 and a SqueezeNet. We used ImageNet pretrained weights but unfreezed the entire network. We used same training settings as for the residual networks. In Figure 1 is depicted training results for the different experiences.

Model	Train Acc	Train Loss	Val Acc	Val Loss
ResNet50	0.992	0.024	0.990	0.041
ResNet34	0.985	0.048	0.980	0.062
ResNet18	0.989	0.033	0.988	0.038
SqueezeNet	0.967	0.101	0.979	0.097
MobileNetv2	0.992	0.022	0.986	0.043

Table 3. Training results for different CNNs after 50 epochs

4.3. Training an auto-encoder CNN architecture

4.3.1 Architecture

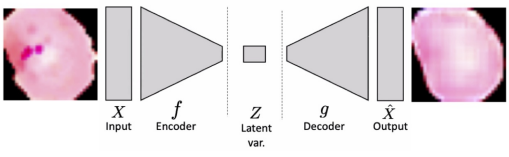

Figure 3. Auto-encoder architecture

In [], an auto-encoder-based architecture is proposed which has the advantage of being efficient and demanding less computationally resources than for a deep network. The idea behind a CNN auto-encoder is that you feed your training data through alternating convolutional and pooling layers into a lower-dimensional latent space representation F, then feed it through convolutional and upsampling layers until you get back to the original size. You train both your encoder $f : E \rightarrow F$ and decoder $g : F \rightarrow E$. The auto-encoder learns to "encode" and "decode" images to get them to be as close as possible to each other.

The output $\hat{X}$ is reconstructed from the input X:

$$\hat{X} = g(f(X))$$

During training we learn both the encoder and the decoder to minimize the distance between the input and the output:

$$\text{f,g} = argmin_{f,g}||X - \hat{X}||^2$$

We try to minimize the MSE error over the problem's parameters θ :

$$\min L_{MSE(}\theta) = ||X - \hat{X}||^2$$

The primal goal of an auto-encoder is dimensional reduction [5]. Here it is used as a classifier: we first train an auto-encoder to learn how to compress and decompress information. Then we will remove the decoder and replace it by fully connected layers which will classify the inputs into the expected classes. We will use the encoder as a feature extractor in a low dimensional space F, and the top layers as a classifier. The advantage of this architecture is it's very small size: it sizes only a few layers.

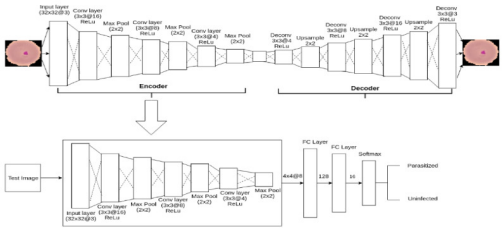

Figure 4. Auto-encoder architecture

The encoder first compresses input X into a latent representation Z:

$$\text{Z} = \text{g(X)}$$

The encoder is composed of three convolutional layers of kernel size 3x3, same padding and stride equal to 1, each followed by a max-pooling layer with window size 2x2. The first convolutional layer has 16 filters, and second and third layers respectively 8 and 4. ReLU activation function, was applied in encoder's hidden units to introduce nonlinearity to the neuron's output.

Figure 5. Inputs versus outputs of the auto-encoder trained model

The decoder reconstructs Z into an output of same size as X.

The decoder is composed of four deconvolutional layers of kernel size 3x3, same padding and stride equal to 1, each followed by an upsampling layer with window size 2x2. The first deconvolutional layer has 4 layers and second, third and forth respectively 8,16 and 3. ReLU activation function was applied to the hidden units to introduce non-linearity.

During testing phase, the decoder is replaced by two fully connected layers in order to detect malaria parasite from a blood smear. Hence, the final model is relatively small (9 layers): it is composed of three convolutional layers, three max pooling layers, one flatten layer, two fully connected layers and a softmax layer which predicts probabilities over the two classes.

4.3.2 Training precisions

At first, the entire auto-encoder was trained on 64 x 64 images over 400 epochs with data augmentation described in Table 1. In order to minimize the MSE loss over the inputs, we used RMSprop as optimizer with a learning rate of 0.001. Figure 5 illustrates the reconstruction of the original inputs.

Image Size	Train Loss	Val Loss
32 x 32	0.097	0.071

Table 4. Training results of the auto-encoder after 400 epochs

Then, we only used the trained encoder structure as feature extractor. Training is done on the fully connected layers during 50 epochs. Binary cross entropy is minimized by using an SGD optimizer with a learning rate and momentum respectively set to 1e-3 and 0.9.

Train Acc	Train Loss	Val Acc	Val Loss
0.672	0.606	0.651	0.631

Table 5. Training results of malaria auto-encoder classifier after 50 epochs

We then tried to train the auto-encoder by unfreezing the encoder's layers. Performance improved a lot with a training validation accuracy above 97%.

Train Acc	Train Loss	Val Acc	Val Loss
0.977	0.073	0.985	0.060

Table 6. Training results of unfreezed malaria auto-encoder classifier after 50 epochs

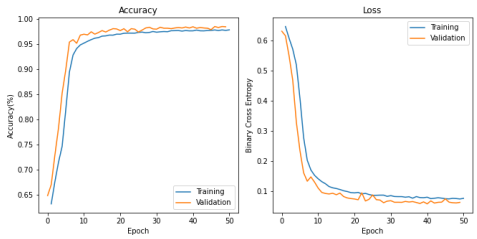

Figure 6. Training results after training malaria classifier during 100 epochs.

5. Evaluation

In this section we evaluate our different models using the following performance metrics : test accuracy (Acc.), f1 score, precision (Prec.), sensitivity (Sens.) and specificity (Spec.). We also considered the size of the model along with performance in order to ensure the viability of our model in terms of computational resources.

Algo	Acc	Loss	F1 Score	Prec	Spec	Sens
ResNet50	0.987	0.039	0.988	0.992	0.984	0.992
Resnet34	0.980	0.056	0.980	0.982	0.983	0.978
Resnet18	0.987	0.038	0.987	0.981	0.982	0.993
Mob.NetV2	0.987	0.042	0.987	0.984	0.985	0.991
SqueezeNet	0.968	0.091	0.968	0.986	0.969	0.986
Autoen.	0.979	0.068	0.980	0.969	0.971	0.989

Table 7. Evaluation of performance metrics for different models on test set

We notice that our best model is ResNet18, with a test accuracy of 0.987 and a loss of 0.038. What is remarkable is

Algo	Image Size	MB	FLOPs
ResNet50	224 x 224	201	4G
ResNet34	224 x 224	118	4G
ResNet18	224 x 224	68	2G
Mob.NetV2	224 x 224	52	579M
SqueezeNet	224 x 224	35	837M
Autoen.	32 x 32	0.5	-

Table 8. Evaluation of computational resources metrics for different models on test set

that it is as efficient as ResNet50 which needs 3 times more memory storage. However, in the medical field, what interests us the most is the proportion of true positives which are well classified, i.e the true positive rate also called sensitivity. We notice that our auto-encoder architecture performs well in terms of sensitivity which is 0.989, somehow close to the deep network's ones. It's lower accuracy is explained by it's higher difficulty to detect the negative class (specificity). Thus if we think back to our original problem, our objective is to build a model which is efficient and demands low computational resources. Table 8 gives us sizes of the different evaluated models. It is clear that the auto-encoder is the best model in terms of demanding energy resources. Moreover, as the input images only sizes 32x32 which is equivalent to 1 Ko storage, we can imagine that a low cost mobile phone could store several hundred of these images. Finally, by it's high sensitivity and small size, we have chosen the auto-encoder, which has a remarkable simple architecture comparing to deeper networks, as our best model.

One of the main criticism of neural networks, is the difficulty to understand what they really do. One way to figure out how they manage to classify our images into two distinct classes, is to find out which pixels from the input are responsible for the predicted class. We can indeed plot activation maps in order to highlight the pixels responsible for the activation in each layer. Let's plot in Figure 7 the activation maps for the final convolutional layer of the auto-encoder.

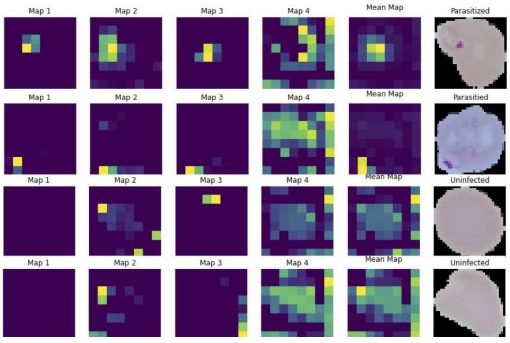

Figure 7. Activation maps for the last convolutional layer for the positive and negative predicted class.

In Figure 7, we observe for the parazitised images, pixels of highest value i.e "activated pixels" localised in the infected zone of the cell. This shows that our classifier has been trained to indeed classify infected pixels in our image as part of a parazitised cell image. Furthermore, we can create a heatmap from the mean activation map in order to overlay the activations on the original image. This technique, allows us to highlight what our model has learned during the training but also to understand cases of misclassification by our model, which is critical in order to improve its performance. In Figure 8, we plot heatmaps for the positive and negative class.

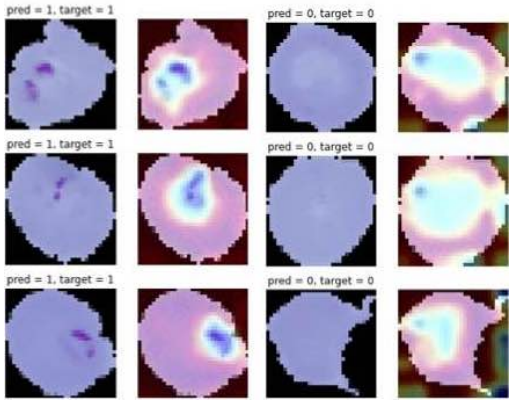

Figure 8. Activation maps for the last convolutional layer for the positive and negative predicted class.

Finally, we can also take a look into the mis-classified images by the auto-encoder in order to understand if our model could do even better. We plot the confusion matrix : false positives and false negatives.

	Uninfected	Parazitised	Support
Uninfected	2518	29	2547
Parazitised	80	2639	2719

Table 9. Confusion Matrix for malaria classifier

We can also look into the mis-classified data's heatmaps, in particular the mis-classified positive class, in order to understand why our model fail to classify some cases.

For most of the false-negatives our model seems to detect infected pixels. But the activations doesn't seem to be important enough in order to consider the image with high certainty as parazitised. Moreover, the model seems to have large activations maps at some zones as if it gave importance to the smooth aspect of the image which characterizes healthy cells.

For most of the false-positives, the model seems to detect small artefacts which makes it think they are infected cells and therefore classifies the image as parazitised. These cases can be quite difficult to learn correctly to the model. The best way to make our model distinguish infected pixels from artefacts would be to learn it the model or train an other model to learn this distinction.

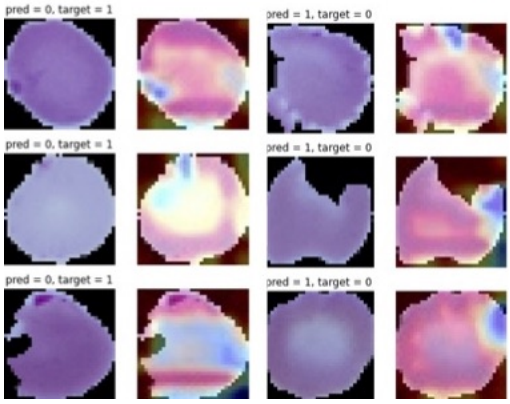

Figure 9. Activation maps for the last convolutional layer for misclassified data.

One way to deal with false negatives versus false positives is to play with the probability acceptance which is now set at 0.5. Lowering the threshold can lead to less false negatives but more false positives. This technique is prevalent in the medical field as we rather prefer being mistaken on a few uninfected patients than infected ones.

When we looked into the mis-classified data, we noticed that most of them are difficult to categorize as 'Parasitized' even for the human eye. This can be due to a camera defect, mislabelled data by the doctor or simply a cell which is very difficult to diagnosis as infected by malaria. With consideration with this issue which often remains in medical imaging classification, we can consider that our model performs well and can be brought to the field to help clinicians and doctors.

6. Discussions and conclusions

After training several models to predict presence of malaria in blood smear images and compared them both in terms of performance and size, we have shown that the auto-encoder classifier suits the best for our problem. We insist that the difficulty of our problem isn't the main classification task, which several teams have performed well, but being efficient in a context of low computational resources. Traditional CNNs may perform well in pattern recognition, but unfortunately they need a lot of memory storage and FLOPs to predict the correct class of an image. This work had the main goal to show that even with a low-resolution image, using an auto-encoder as feature extractor helps us to design a simple and small model. By computing features in a latent representation space of low-dimension, we managed to compress the dimensions of our problem. Then by using a simple classifier upon this feature extractor we came up with an accuracy of 0.98 which is remarkable regarding the size of the model and the inference time (less than 1s). If in 2018, Rajaraman et al. proposed an efficient solution it wasn't viable to the problem they were trying to answer. The paper Fuhad et al. wrote in 2020, brings a smart and simple solution which underlines the importance of understanding the context of problems and being able to rethink traditional deep learning techniques which may sometimes be forgotten.

References

[1]Fact Sheet about Malaria. [(accessed on 29 December 2019)];2019 Available online:https://www.who.int/news-room/fact-sheets/detail/malaria

[2] Chunwei Xia, Jiacheng Zhao, Huimin Cui, Xiaobing Feng, and Jingling Xue. 2019. DNNTune: Automatic Benchmarking DNN Models for Mobile-cloud Computing. ACM Trans. Archit. Code Optim. 16, 4, Article 49 (January 2020). DOI:https://doi.org/10.1145/3368305

[3] 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;9:e104855. [PMC free article] [PubMed] [Google Scholar]

[4] 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. [Google Scholar]

[5] Liang Z., Powell A., Ersoy I., Poostchi M., Silamut K., Palaniappan K., Guo P., Hossain M.A., Sameer A., Maude R.J., et al. Cnn-based image analysis for malaria diagnosis; Proceedings of the 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM); Shenzhen, China. 15–18 December 2016; pp. 493–496.

[6] Rajaraman S., Antani S.K., Poostchi M., Silamut K., Hossain M.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. doi: 10.7717/peerj.4568. [PMC free article] [PubMed] [CrossRef] [Google Scholar]

[7] Fuhad KMF, Tuba JF, Sarker MRA, Momen S, Mohammed N, Rahman T. Deep Learning Based Automatic Malaria Parasite Detection from Blood Smear and its Smartphone Based Application. Diagnostics (Basel). 2020;10(5):329. Published 2020 May 20. doi:10.3390/diagnostics10050329

[8] Corrected Malaria Data-Google Drive. [(accessed on 29 December 2019)];2019 Available online: https://drive.google.com/drive/folders/10TXXa6B_D4AKuBV085tX7UudH1hINBRJ?usp=sharing.