EfficientDet-based Deep Learning Model for Detection and Classification of Malaria-Infected Human Blood Smears: An End-to-End Solution

Mohammad Mahdi Heydari

March 10, 2023

1 Abstract

Here we try to tackle the cell segmentation problem on the BBBC041[1] dataset. The dataset's images contains 6 classes of cells in the blood stream and the goal is to find the cells coordinate and class for every cell which is present in the image. To accomplish this, a recently developed model named EfficientNet by google's brainAI team is utilized. By using this, the trained model succeeded to achieve map 52 over the test dataset.

2 Dataset

Images in this dataset contains 6 classes from two categories (infected and uninfected). for each image provided the annotation data which is bounding box coordinates and class for each cell. Although the author mentions that there exist only 6 classes, but a 7th class named "difficult" observed in the data which we ignored for training. Figure 1 shows the class frequencies for each set.

To shrink the problem space, we added an extra preprocessing step and that would be removing the background using Otsu's method which is a automatic image thresholding technique. after downsclaing images to the network input size (512×512 px) we perform otsu thresholding to find a binary mask over the cells. to make the mask smoother, we used morphological operations like binary_closing and binary_fill_holes in order to keep entire cell area in image. for the final step, by dividing all pixel values to 255 the network input well always be a tensor sized 512×512×3 which it values is between 0 and 1 . Figure 2 shoes the output of preprocessing step for some of training samples.

3 Related Works

There are several Deep Leaning models for this task like Yolo, FasterRCNN and ... in this work we used a recently proposed model by Google brainAI team named EfficientDet.

[1] malaria infected human blood smears: https://bbbc.broadinstitute.org/BBBC041

Figure 1: Class frequencies for dataset.

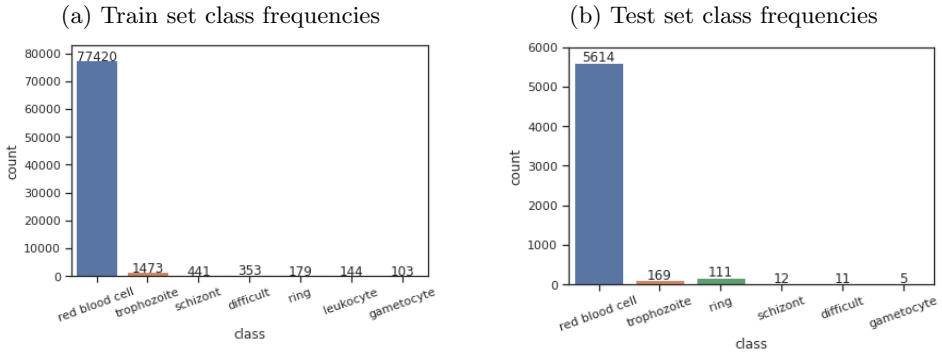
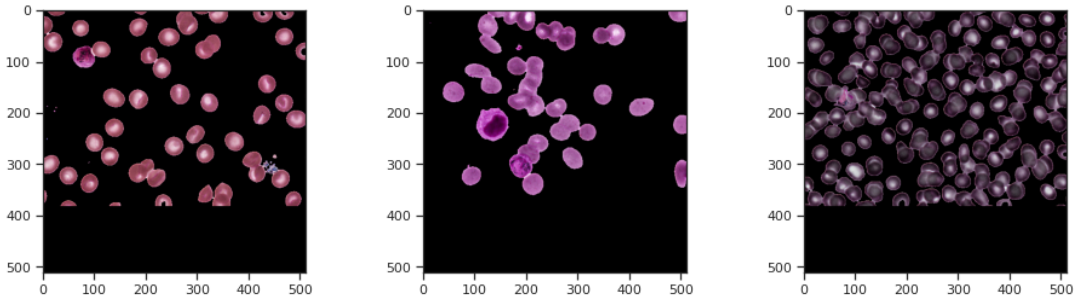


Figure 2: Output of preprocessing step.

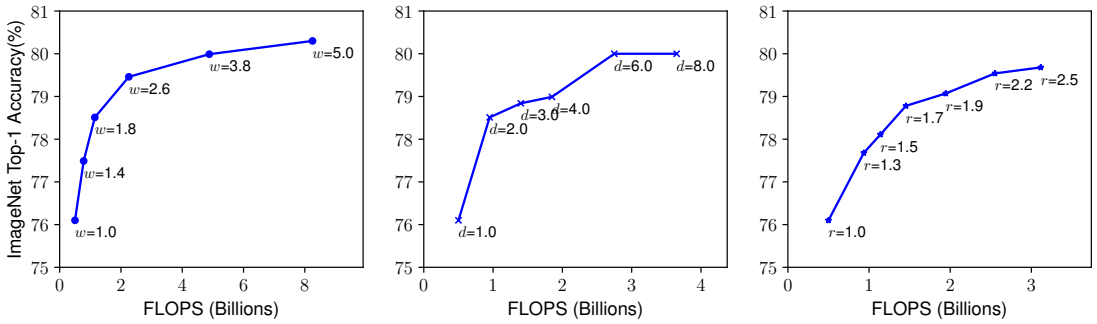


it's an scalable and efficient one-stage multi-scale object detector which achieved state-of-the-art results on Microsoft's COCO dataset.

EfficientDet is based on their previous work EfficientNet. the EfficientNet is a classifier model which built on famous MobileNet architecture. as the paper title suggests (Rethinking Model Scaling for Convolutional Neural Networks) they parameterized model's computational cost (flops) using number of layers, number of channels in each layer and input resolution of model. they showed that increasing number of layers or channels independently can increase the model accuracy to a point and after that accuracy will saturate (Figure 3). so they relate this three hyperparameter and optimized model's computational cost subject to them. they introduced ϕ parameter as below:

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{1}$$

Figure 3: Effect of increasing number of layers, channels and input resolution on model's accuracy



so increasing ϕ will give us a more deeper and wider model which uses a higher resolution input. Result's on ImageNet dataset shows that it's actually quite effective in compare to the most recent methods (Figure 4).

4 Poposed Model

Following the EfficientDet architecture, we used EfficientNet $\phi = 0$ as backbone network which include multiple MBConvBlock layers to extract the feature map. Table 1 shows the overall properties of the blocks at the backbone network. we use level 3-7 level features from the backbone network and feed them into to BiFPN layers. this is where this multi-scale features fuse together with a top-down and bottom-up bidirectional feature fusion operation. then we feed them to box and class prediction networks to get the final result. Figure 5 shows the overall architecture of EfficientDet.

4.1 MBConvBlock

MBConvBlock is the main building block of the backbone network which first introduced in MobileNet V2. each block contains this phases (copied from the paper):

- **Expansion phase:** we will expand our layer and make them wide as mentioned in Inverted residual block (connected blocks are narrow and inner blocks are wider, here we are making layer wider just by increasing the number of channels).
- **Depthwise convolution phase:** after expansion, we perform depthwise convolution with kernel size mentioned in block argument.
- **Squeeze and excitation phase:** now, we extract global features with global average pooling and squeeze numbers of channels. the output of this phase can be considered as weightage for each channel in the output.

Figure 4: Relation of ϕ with model's accuracy (B0 to B6)

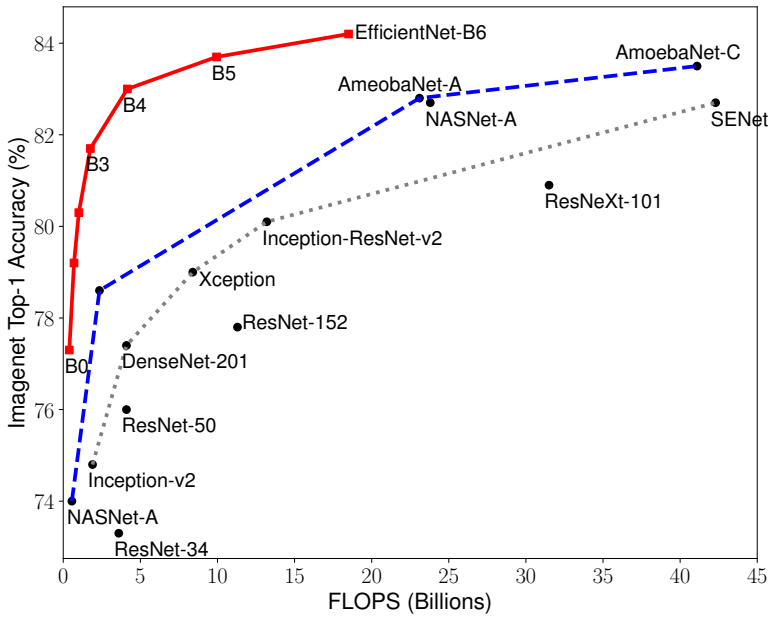
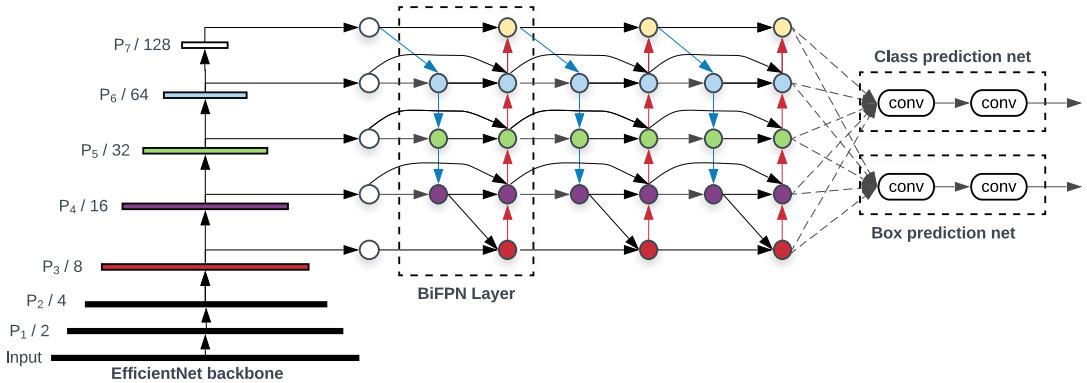


Figure 5: EfficientDet architecture



- **Output phase:** after the squeeze and excitation, we apply convolution operation that gives output filters mention in the argument block.

4.2 Swish Activation

to resolve the ReLu's problem with negative values brainAi team introduced a new activation function named Swish:

$$Swish(x) = x * sigmoid(x) \tag{2}$$

they proved using swish activation function the $InceptionResNetV2$ achieve 0.6% more accuracy on the ImageNet dataset.

4.3 BiFPN network

this layer doing bidirectional cross-scale connections and weighted feature fusion and originally introduced in the efficientDet's paper. given a list of multi-scale features $\vec{P}^{in} = (P^{in}_{l_1}, P^{in}_{l_2}, ...)$, the goal is to find a transformation f that can effectively aggregate different features and output a list of new features: $\vec{P}^{out} = f(\vec{P}^{in})$. in this work we implemented *Fast normalized fusion* which introduced in the paper. $O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i$, where $w_i \geq 0$ is ensured by applying a Relu after each w_i, and $\epsilon = 0.0001$ is a small value to avoid numerical instability. for example here we describe the two fused features at level 6 for BiFPN:

$$P^{td}_6 = Conv\left(\frac{w_1 \cdot P^{in}_6 + w_2 \cdot Resize(P^{in}_7)}{w_1 + w_2 + \epsilon}\right)$$
$$P^{out}_6 = Conv\left(\frac{w'_1 \cdot P^{in}_6 + w'_2 \cdot P^{td}_6 + w'_3 \cdot Resize(P^{out}_5)}{w'_1 + w'_2 + w'_3 + \epsilon}\right)$$

where P^{td}_6 is the intermediate feature at level 6 on the top-down pathway, and P^{out}_6 is the output feature at level 6 on the bottom-up pathway. All other features are constructed in a similar manner. (this part is copied from the original paper)

4.4 Focal Loss

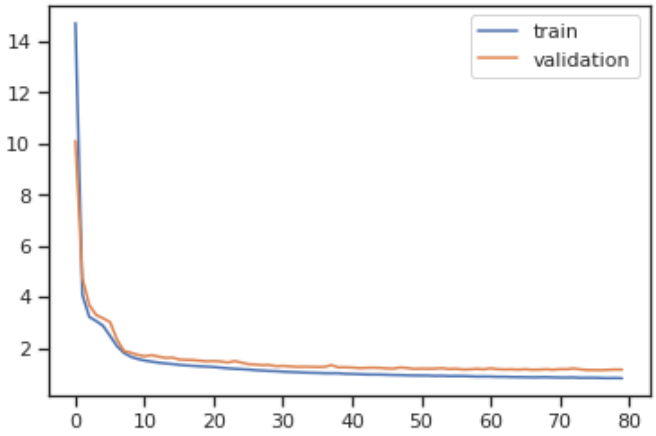
duo to huge the amount of pixel belongs to the background in compare of pixels which belongs to the actual objects, there is a large imbalance between class frequencies. to solve that we used Focal Loss with below formula to assign bigger penalty to the model for misclassifying the object's pixels:

$$FL(P_t) = -(1 - P_t)^{\gamma} Log(P_t) \tag{3}$$

Table 1: Backbone layers

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	stride	#Layers $\hat{L}_i$
-	Conv3x3	256×256	32	2	1
-	MBConv1, k3x3	256×256	16	1	1
-	MBConv6, k3x3	128×128	24	2	2
P_3	MBConv6, k5x5	64×64	40	2	2
-	MBConv6, k3x3	32×32	80	2	3
P_4	MBConv6, k5x5	32×32	112	1	3
-	MBConv6, k5x5	16×16	192	2	4
P_5	MBConv6, k3x3	16×16	320	1	1
-	Conv3x3	16×16	64	2	1
P_6	MaxPool3x3	8×8	64	2	1
P_7	MaxPool3x3	4×4	64	2	1

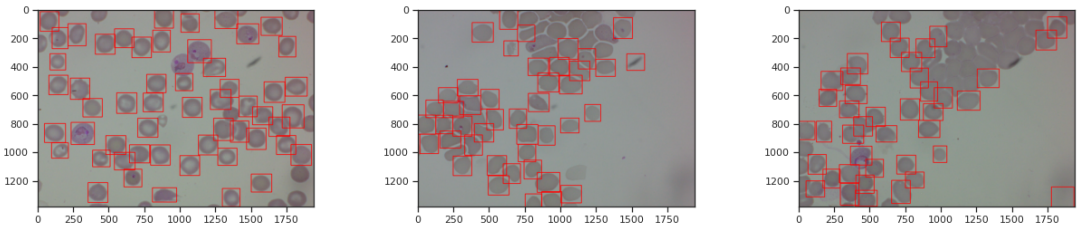
Figure 6: Model's loss along training



5 Results

we trained the model with 100 epochs with Adam optimizer (10^{-4} learning rate) which took near 3 hours on the 80% random part of the train set and used the other 20% as validation data. Figure 6 shows the loss of the model for train and validation set during the training process. Figure 7 shows the final results on some of the test images. we clearly can see the model does a perfect job in locating the cells, the classification results are so bad that isn't worth mentioning.

Figure 7: Output of trained model on the test set.



6 Discussion

Speaking around the poor performance of the model, it did a perfect job in predicting bounding boxes but the classification result is worthless, the output is "red blood cell" for every thing. one reason for this is a huge class imbalance which in this context can't be solve with data augmentation or bootstrapping. one way to solve this issue is using the α parameter in the focual loss, although we tried playing around this but didn't help much in the first shot and need more work to solve the issue.

Using one-stage detectors are fairly rare in this context which i believe mostly is because the common pattern in cell shapes makes it easy for two-stage detectors. Although the EfficientDet architecture is the state-of-the-art result holder for 90-class COCO dataset, it did poor job solving this problem. it's good to mention that the thresholding method helped model by a lot but it is not a common and standard procedure using this kind of models. this model is designed to be an end-to-end solution.

Our intent in using this architecture was try to implement a new state-of-the-art model and after spending a lot of time it was unpleasant to ignore it.

Reference

Lin, Tsung-Yi et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

Sandler, Mark et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV].

Tan, Mingxing and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].

Tan, Mingxing, Ruoming Pang, and Quoc V. Le. *EfficientDet: Scalable and Efficient Object Detection*. 2020. arXiv: 1911.09070 [cs.CV].