Collabits Journal
Vol 1 No. 1 January 2024 : 50-54
E-ISSN : 3046-6709
https://publikasi.mercubuana.ac.id/index.php/collabits

The Implementation of *Residual Network* with *Monte Carlo Dropout* for Predicting Malaria through Thin Blood Smear Images

Siti Maesaroh[1], Lukman Hakim[2], Ita Erliyani[3*], Khairul Hidayat[4]
[1,2,4] Computer Science study program, Universitas Mercu Buana, Indonesia
[3] System Information Study Program, University of Raharja, Indonesia

*Coressponden Author: ita.erliyani@raharja.info

Abstract - *Malaria is a sudden-onset febrile illness caused by the Plasmodium parasite, which is transmitted to humans through the bite of female Anopheles mosquitoes. When mosquitoes bite humans, the parasites enter the human body and establish themselves in the liver. The similarity of its symptoms to common illnesses makes it challenging to identify malaria without microscopic examination of blood smears. However, the accuracy of microscopic examination relies on the quality of the smears and the expertise in classifying and counting infected and uninfected cells. Conducting such examinations on a large scale can be difficult and may lead to suboptimal results. To address these limitations, a deep learning approach known as the Residual Network can be employed. The Residual Network is a Convolutional Neural Network architecture that enables the network to skip or bypass specific layers. Skip connections facilitate a more efficient gradient flow during training and enable the network to acquire improved representations of novel data. In order to make the model adaptable to variations in the data during training, the model incorporates the Monte Carlo dropout method. This method helps prevent the network from becoming excessively tailored to specific training examples, thereby enhancing the model's ability to generalize. Through the utilization of the ResNet architecture and Monte Carlo dropout, the model is capable of diminishing the loss rate as the training process advances. Even with 35 training iterations and a batch size of 32, the model is able to attain an accuracy rate of 97% and a loss rate of 6.5%.*

Keywords :
Residual Network;
Monte Carlo Dropout;
Deep Learning;
Convolutional Neural Network;
Medical image analysis;

Article History:
Received: 28-10-2023
Revised: 03-12-2023
Accepted: 25-12-2023

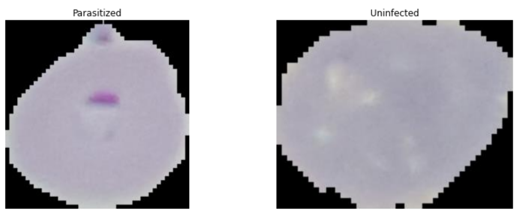
Article DOI : 10.22441/collabits.v1i1.25564

1. INTRODUCTION

Malaria is a sudden-onset febrile disease caused by the Plasmodium parasite, which is transmitted to humans through the bite of a female *Anopheles mosquito*. When a mosquito bites a human, the parasite enters the human body and resides in the liver [1]. The initial symptoms of malaria typically manifest 10-15 days following the mosquito bite, with the commonly observed symptoms being fever, headache, and chills.[2] Due to the similarity of these symptoms with common illnesses, identifying them as malaria can be challenging. Microscopic examination of the patient's blood smear has emerged as the gold standard for malaria diagnosis because this method is both rapid and cost-effective[3]. Nonetheless, the accuracy of microscopic examination relies on the quality of the smear and the expertise in classifying and counting parasitized and uninfected cells[1] In a journal authored by Afrizal Zein, the Python programming language was employed along with the ResNet model, employing a methodology similar to the division of data into three categories: training, testing, and validation. However, during the training data process, researchers utilized the 'hard' library available in Python for a total of 50 iterations, yielding an accuracy rate of 96.5% in training, 96.78% in validation, and 97% in testing [4]. This study will employ Python and build upon prior research using a dataset comprising 13,780 images each of infected and uninfected thin blood smears. Subsequently, the training process will incorporate the Monte Carlo dropout method with a dropout rate of 45% and will consist of 35 iterations to investigate whether there is any impact on improving accuracy compared to previous research.

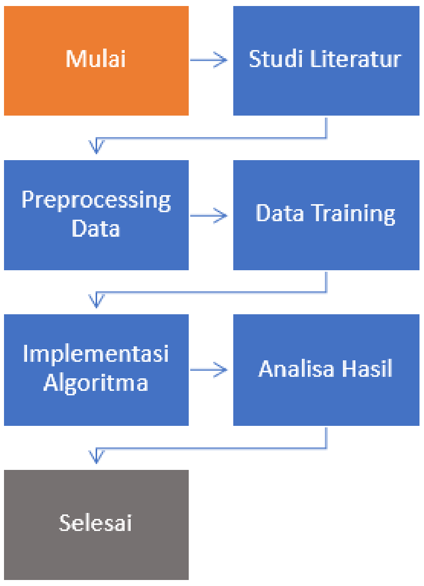
smears labeled as '*uninfected*.'

Figure 1. Infected data samples on the left and uninfected on the right



The stages of research to be conducted are outlined in the research framework below:

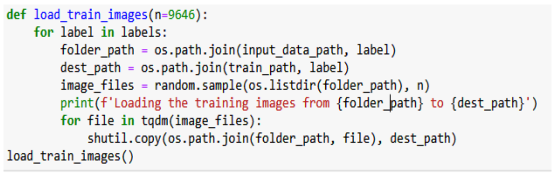
Figure 2. Research Stages



A. *Pre-Processing*

The pre-processing stage divides the dataset into three parts, namely *training*, *validation*, and *testing*, while the dataset ratio is divided into 70% for *training*, 20% for *validation*, and 10% for *testing* using python with the code in Figure 3.

Figure 3. Programme code for split dataset



```
def load_train_images(n=9646):
    for label in labels:
        folder_path = os.path.join(input_data_path, label)
        dest_path = os.path.join(train_path, label)
        image_files = random.sample(os.listdir(folder_path), n)
        print(f'Loading the training images from {folder_path} to {dest_path}')
        for file in tqdm(image_files):
            shutil.copy(os.path.join(folder_path, file), dest_path)
load_train_images()
```

The load_train_images function has the purpose of loading the training images of each class and saving them to the appropriate destination folder. It accepts an optional parameter '*n*' that regulates the quantity of images to be loaded for each class, and the training data necessitates 9646 samples for each label. The algorithm in this code is employed iteratively to partition the data into 2756 samples for validation and 1378 samples for testing for each label.

B. Normalisation and Augmentation

Normalization is employed to enhance numerical stability, expedite algorithm convergence, eliminate bias or scale disparities, and simplify data interpretation and comparison. In the realm of image processing, normalization additionally aids in mitigating variations in light intensity, contrast, or color scale across different images.

Data augmentation, on the other hand, is a method employed to increase the volume of training data by creating new variations of the existing data. This is achieved by applying basic transformations or manipulations to the images within the dataset. The objective of data augmentation is to enhance the model's generalization capability and mitigate overfitting.

At this point, the image dimensions are resized to 64x64 to expedite the *training process*, maintaining an image size that is not excessively large yet remains consistent with the original image. The configuration for data augmentation can be found in Table 1.

Table 1. Augmentation Configuration

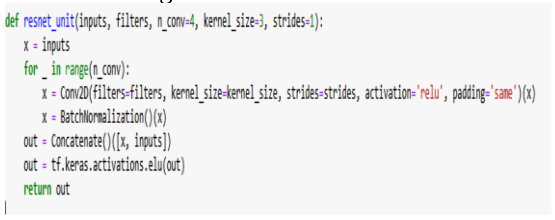
Parameters	Configuration
Horizontal_Flip	True
Rotation_range	30
Zoom_range	0.2

C. *Residual Network*

Residual Network or ResNet is an architecture for CNN algorithms developed by researchers from Microsoft Research. It is a method of adding *residual*, or skip, connections that directly adds the output of each layer to the *output of the* next layer. This method was formed to overcome one of the problems in deep learning, namely exploding and vanishing gradient.[5].

Residual Network (ResNet) uses *residual blocks* that allow the network to *skip* some layers, *skip connections* allow more efficient gradient flow during training and allow the network to learn better representations of new data. The programme code for building the ResNet architecture can be seen in Figure 14.

Figure 4. Code for ResNet units



```
def resnet_unit(inputs, filters, n_conv=4, kernel_size=3, strides=1):
    x = inputs
    for _ in range(n_conv):
        x = Conv2D(filters=filters, kernel_size=kernel_size, strides=strides, activation='relu', padding='same')(x)
        x = BatchNormalization()(x)
    out = Concatenate()([x, inputs])
    out = tf.keras.activations.elu(out)
    return out
```

The function 'resnet_unit' is responsible for constructing a unit block within the ResNet architecture. This unit block comprises several convolutional layers, batch normalization, and skip connections. This function takes as input the input tensor of the previous layer, the number of *filters* for the convolution layer, and some other parameters such as the number of convolution layers (*n_conv*), kernel size (*kernel_size*), and *strides*. This function returns the *output* which is the result of the ResNet unit block.

Here is the algorithm flow for resnet_unit:

1. Initialises the variable x with the inputs value.
2. Iterate *n_conv* times.
   a. Apply a convolution layer with the *same* number of filters, kernel size, step, ReLU activation function, and padding to x.
   b. Perform batch *normalisation* on x using *BatchNormalisation*.
3. Combine the output x with the initial inputs inputs using *Concatenate*.
4. Apply the ELU activation function to the previous combined results.
5. Returns the final result out.

After creating the ResNet unit, this function will be integrated into the model that will be constructed as the

ResNet architecture, which will serve as the model.

D. *Monte Carlo Dropout*

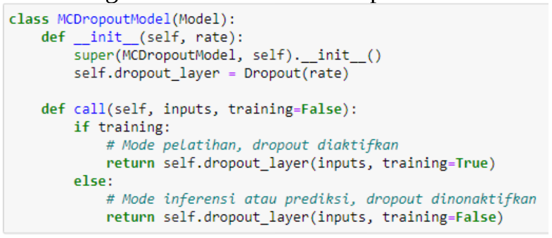
Dropout is typically a technique employed to mitigate overfitting in neural networks. This approach operates by deactivating neurons based on a predetermined rate during training, but once training is completed, all neurons remain active *[6]*.

Monte Carlo Dropout is a technique introduced by Gal and Ghahramani. They conceptualized dropout as a sampling method, which is equivalent to estimating the uncertainty of a deep Gaussian process. This dropout technique provides a consistent way to generate model uncertainty estimates. Monte Carlo Dropout is a method employed to combat overfitting during the training of a model. During the testing phase of Monte Carlo Dropout, stochastic dropout is applied, which involves randomly dropping out some units with a fixed probability, and this process is repeated. Predictive results are obtained by re-inferencing several times and combining the results. This technique produces more stable forecasts and accounts for model uncertainty. It prevents the network from becoming too specialised to specific training examples and improves the generalisation of the model.

Monte Carlo Dropout has discovered numerous practical applications, such as in time series prediction and medical imaging.[6]

In this study, the development of the Monte Carlo Dropout function for the ResNet architecture is outlined as follows:

Figure 4. Monte Carlo Dropout Code



```
class MCDropoutModel(Model):
    def __init__(self, rate):
        super(MCDropoutModel, self).__init__()
        self.dropout_layer = Dropout(rate)

    def call(self, inputs, training=False):
        if training:
            # Mode pelatihan, dropout diaktifkan
            return self.dropout_layer(inputs, training=True)
        else:
            # Mode inferensi atau prediksi, dropout dinonaktifkan
            return self.dropout_layer(inputs, training=False)
```

Below is the program flow of the Monte Carlo Dropout algorithm:

a) Model initialisation: *The MCDropoutModel* is created by initialising a dropout layer object with the specified dropout level.
b) Training mode: When the model is called with the *training=True* parameter, the training mode is activated.
c) Active dropout: In training mode, the dropout layer is applied to the *inputs* using the *dropout_layer(inputs, training=True)* method. This activates dropout, where each input element has a randomised chance *rate of being* dropped out.
d) Inference mode: When the model is called with the *training=False* parameter, inference or prediction mode is enabled.
e) Dropout is disabled: In inference mode, dropout is not applied to the input, so the *dropout_layer(inputs, training=False)* method will return the input unchanged.

In training mode, dropout is applied to the input to facilitate regularization and diminish overfitting. In inference mode, dropout is deactivated to obtain stable predictions and estimate prediction uncertainty using the Monte Carlo Dropout method.

2. LITERATURE REVIEW

The following is a table of research related to research using the ResNet model

1. Implementation of Convolutional Neural Networks for Batik Image Dataset, 2022. The CNN model is built using convolutional and pooling layers to extract significant features from batik images, achieving an accuracy rate of 95%. Furthermore, the model effectively addresses the challenge of classifying batik images with variations in scale, rotation, and distortion.

2. Vehicle detection using improved region convolution neural network for accident prevention in smart roads, 2022. The proposed approach integrates the utilization of a Region Proposal Network (RPN) with a Convolutional Neural Network (CNN) to enhance the accuracy of vehicle detection. The authors conducted an assessment on a vehicle dataset and observed that the proposed model achieved a commendable detection accuracy rate of 98.68%, while maintaining a processing speed of 30% per second.

3. Li-ion battery degradation modes diagnosis via Convolutional Neural Networks, 2022. The authors present an approach that employs two distinct CNN architectures for feature extraction from battery degradation data. Subsequently, a logistic regression model is utilized to predict potential degradation modes. During cross-validation, the CNN model utilizing the ResNet-50 architecture demonstrates superior performance in predicting degradation modes with an accuracy of 95.1%, surpassing the CNN model utilizing the VGG-16 architecture, which achieved an accuracy of 91.6%.

4. In 2022, the prediction of COVID-19 infection from CT scan lung images was conducted using the Iterative Convolutional Neural Network (CNN) model. The method underwent evaluation using a dataset of CT images gathered from various medical centers and achieved impressive results in COVID-19 infection prediction. The training process involved two models, CNN1 with 2 *hidden layers* and CNN2 with 3 hidden layers. Both CNN1 and CNN2 were trained with 100 datasets, and they exhibited higher accuracy in COVID-19 classification compared to other models. The CNN2 model achieved the highest classification accuracy of 89% during the 5th iteration.

5. Deep Convolution Neural Network sharing for the multi-label images classification, 2022. divides the image data into a series of sub-images and then trains each DCNN (Deep Convolutional Neural Network) on a subset of the sub-images Each DCNN is trained to identify a specific subset of labels, and the outputs of all DCNNs are then combined to make the final classification decision. The multi-feature network achieved the highest classification score

compared to the deep neural network with 83.31% f1 score for the Amazon rainforest dataset. The f1 score values were 88.81% for the Pascal VOC 2007 dataset, 87.71% for Nuswide, and 88.64% for Pascal VOC 2012.

3. RESULTS AND DISCUSSION

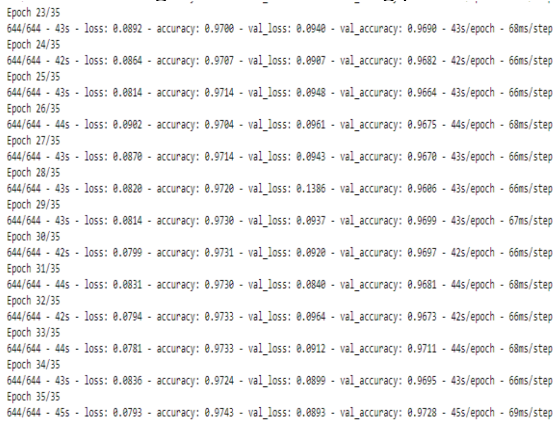
The architecture employed in this study comprises various types of interconnected layers. The input of this model is an image with a size of 64x64 pixels and 3 colour channels, the first layer in the model is a 2-dimensional *Convolutional* with 8 filters with a size of 3x3 pixels. After that, *batch normalisation is* performed to normalise the values in the layer. This process is repeated 4 times with the same number of filters at each step, after which, the results of the *Convolutional* layer and *batch normalisation are* combined with the initial input using the *concatenate* operation. The results are then run through the ELU activation function and *average pooling* with a size of 2x2 pixels is performed.

This process is followed by *Convolutional* layers, *batch normalisation, concatenation,* and *average pooling* which are performed several times with an increasing number of filters at each step. In the next stage, the results of the previous process are run through several *convolutional* and *batch normalisation* layers with an increasing number of filters. Afterwards, *concatenation* and *average pooling are performed*. The above steps are repeated with a larger number of filters until the final result is obtained with a size of 2x2 pixels and 507 *channels*. The result is then flattened into a 1-dimensional vector and connected to a *dense* layer with 4096 units. There is also a mc_dropout_model layer which is a modification of the *dense layer* that uses the dropout method to reduce *overfitting*, finally, there is a final Dense layer with 4096 units which is the output of this model.

In this training, the *batch size* used was 32 with a total of 35 iterations to determine how much improvement in *loss* and accuracy compared to validation. Batch size is the number of data samples used in one training iteration in the model training process.

The results of the model training iterations can be seen in Figure 6.

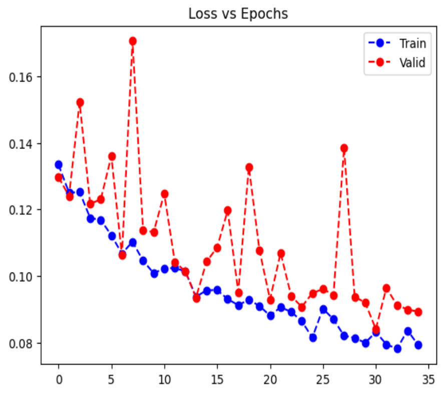
Figure 5. Model training process



```
Epoch 23/35
644/644 - 43s - loss: 0.0892 - accuracy: 0.9700 - val_loss: 0.0940 - val_accuracy: 0.9690 - 43s/epoch - 68ms/step
Epoch 24/35
644/644 - 42s - loss: 0.0864 - accuracy: 0.9707 - val_loss: 0.0907 - val_accuracy: 0.9682 - 42s/epoch - 66ms/step
Epoch 25/35
644/644 - 43s - loss: 0.0814 - accuracy: 0.9714 - val_loss: 0.0948 - val_accuracy: 0.9664 - 43s/epoch - 66ms/step
Epoch 26/35
644/644 - 44s - loss: 0.0902 - accuracy: 0.9704 - val_loss: 0.0961 - val_accuracy: 0.9675 - 44s/epoch - 68ms/step
Epoch 27/35
644/644 - 43s - loss: 0.0870 - accuracy: 0.9714 - val_loss: 0.0943 - val_accuracy: 0.9670 - 43s/epoch - 66ms/step
Epoch 28/35
644/644 - 43s - loss: 0.0820 - accuracy: 0.9720 - val_loss: 0.1386 - val_accuracy: 0.9606 - 43s/epoch - 66ms/step
Epoch 29/35
644/644 - 43s - loss: 0.0814 - accuracy: 0.9730 - val_loss: 0.0937 - val_accuracy: 0.9699 - 43s/epoch - 67ms/step
Epoch 30/35
644/644 - 42s - loss: 0.0799 - accuracy: 0.9731 - val_loss: 0.0920 - val_accuracy: 0.9697 - 42s/epoch - 66ms/step
Epoch 31/35
644/644 - 44s - loss: 0.0831 - accuracy: 0.9730 - val_loss: 0.0840 - val_accuracy: 0.9681 - 44s/epoch - 68ms/step
Epoch 32/35
644/644 - 42s - loss: 0.0794 - accuracy: 0.9733 - val_loss: 0.0964 - val_accuracy: 0.9673 - 42s/epoch - 66ms/step
Epoch 33/35
644/644 - 44s - loss: 0.0781 - accuracy: 0.9733 - val_loss: 0.0912 - val_accuracy: 0.9711 - 44s/epoch - 68ms/step
Epoch 34/35
644/644 - 43s - loss: 0.0836 - accuracy: 0.9724 - val_loss: 0.0899 - val_accuracy: 0.9695 - 43s/epoch - 66ms/step
Epoch 35/35
644/644 - 45s - loss: 0.0793 - accuracy: 0.9743 - val_loss: 0.0893 - val_accuracy: 0.9728 - 45s/epoch - 69ms/step
```

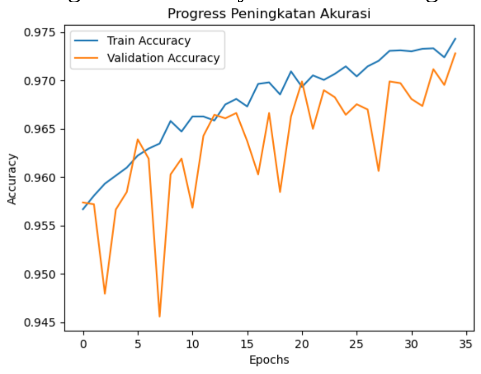
After the training results are obtained, the next step is data visualisation to see the development of the model during the training process.

Figure 6. Plot of loss values during training



The training *loss* decreased as the number of *epochs* increased, indicating that the model was able to reduce the prediction error. Although there are small fluctuations, the overall validation *loss* tends to be stable and indicates that the model is not overfitting.

Figure 7. Accuracy value on training



The training accuracy increases with the number of epochs, reaching about 97% at the end of training. This shows that the model successfully learnt the patterns present in the training data. The validation accuracy also increases and reaches about 97%, indicating that the model can make good predictions on data that has never been seen before.

4. CONCLUSION

After the model was formed and evaluated, the results of the *ResNet* model research with *Monte Carlo dropout,* several conclusions were obtained.

1. The training accuracy increases with the number of epochs, reaching about 97% at the end of training. This shows that the model successfully learnt the patterns present in the training data.
2. The training *loss* decreased as the number of *epochs* increased, indicating that the model was able to reduce the prediction error.
3. The validation accuracy also improved and reached around 97%, indicating that the model can perform well on data that has never been seen before.

4. There are small fluctuations, the overall validation *loss* tends to be stable and indicates that the model is not overfitting.
5. This model can be used to predict malaria in thin blood smear images.

5. ADVICE

Based on the results of the model training, there are several suggestions that can be given in the form of:

1. Even though the model's accuracy on the training and validation datasets is already relatively high, there may still be opportunities to enhance the model's performance. One approach involves fine-tuning the model parameters, such as the *optimizer*, *learning rate*, or network architecture. By conducting additional experiments, it is feasible to identify a more optimal configuration that can further improve accuracy and minimize model loss.
2. The dataset utilized in this study was not directly obtained from a hospital source. Therefore, the model needs to undergo training using a dataset directly acquired from hospitals with a significant number of malaria cases.

BIBLIOGRAPHY

[1] W. Kusuma, A. A. W. Lestari, S. Herawati, W. Putu, and S. Yasa, "Microscope Examination and Rapid Diagnostic Tests in Establishing Malaria Diagnosis," 2014. doi: 10.24843/eum..

[2] World Health Organisation, "Malaria," Jul. 26, 2022. https://www.who.int/news-room/fact-sheets/detail/malaria (accessed Nov. 20, 2022).

[3] N. Ritung, V. D. Pijoh, and J. B. B. Bernadus, "Comparison of Rapid Diagnostic Test (RDT) Effectiveness with Microscope Examination in Patients with Clinical Malaria," Manado, Jul. 2018.

[4] A. Zein, "Detection of Malaria Using Medical Images Analysis with Deep Learning Python," 2019.

[5] B. Zeng, "Towards Understanding Residual Neural Networks," 2019. Accessed: Oct. 18, 2022. [Online]. Available: https://hdl.handle.net/1721.1/123067

[6] A. Labach, H. Salehinejad, and S. Valaee, "Survey of Dropout Methods for Deep Neural Networks," Apr. 2019, [Online]. Available: http://arxiv.org/abs/1904.13310

[7] S. Coulibaly, B. Kamsu-Foguem, D. Kamissoko, and D. Traore, "Deep Convolution Neural Network sharing for the multi-label images classification," *Machine Learning with Applications*, vol. 10, p. 100422, Dec. 2022, doi: 10.1016/j.mlwa.2022.100422.

[8] M. M and S. P, "COVID-19 infection prediction from CT scan images of lungs using Iterative Convolution Neural Network model," *Advances in Engineering Software*, vol. 173, Nov. 2022,doi: 10.1016/j.advengsoft.2022.103214

[9] V. Ayumi, I. Nurhaida, and H. Noprisson, "Implementation of Convolutional Neural Networks for Batik Image Dataset," 2022.

[10] B. Zeng, "Towards Understanding Residual Neural Networks," 2019. Accessed: Oct. 18, 2022. [Online]. Available: https://hdl.handle.net/1721.1/123067

[11] P. Pansawira, "Malaria - Symptoms, causes and treatment - Alodokter," Apr. 11, 2022. https://www.alodokter.com/malaria (accessed Nov. 20, 2022).

[12] Y. Djenouri, A. Belhadi, G. Srivastava, D. Djenouri, and J. Chun-Wei Lin, "Vehicle detection using improved region convolution neural network for accident prevention in smart roads," *Pattern Recognit Lett*, vol. 158, pp. 42-47, Jun. 2022, doi: 10.1016/j.patrec.2022.04.012.