Journal Sensi

Print ISSN: 2461-1409
Online ISSN: 2655-5298

Research Article **Open Access (CC–BY-SA)**

Penerapan *Residual Network* Dengan *Monte Carlo Dropout* Untuk Prediksi Malaria Melalui Citra Hapusan Darah Tipis

Siti Maesaroh [*1], Ita Erliyani [2], Muhammad Zakki Mardhi [3]
[1, 3] Program Studi Teknik Informatika, Fakultas Ilmu Komputer Universitas Mercu Buana, Jakarta
[2]Program Studi Sistem Informasi, Fakultas Sains dan Teknologi Universitas Raharja
e-mail:[*1]siti.maesaroh@mercubuana.ac.id , [*2]ita.erliyani@raharja.info,
[*3]41519010126@student.mercubuana.ac.id

Abstrak

Malaria adalah penyakit demam akut yang disebabkan oleh parasit Plasmodium, yang ditularkan ke manusia melalui gigitan nyamuk Anopheles betina. Ketika nyamuk menggigit manusia, parasit akan memasuki tubuh manusia dan bersarang di hati. dikarenakan gejala nya serupa dengan penyakit umum membuat malaria sulit dikenali tanpa pemeriksaan mikroskopik pada hapusan darah. Namun, akurasi pemeriksaan mikroskop tergantung pada kualitas hapusan, keahlian dalam mengklasifikasikan dan menghitung sel yang diparasit dan tidak terinfeksi. Pemeriksaan seperti itu bisa sulit untuk diagnosis skala besar dan mengakibatkan kualitas yang buruk, untuk menutupi kekurangan tersebut dapat digunakan suatu metode dalam deep learning berupa Residual Network. Residual network merupakan salah satu arsitektur dari model Convolutional Neural Network yang memungkinkan jaringan untuk melompati atau skip beberapa lapisan, skip connection memungkinkan aliran gradien yang lebih efisien selama pelatihan dan memungkinkan jaringan untuk belajar representasi yang lebih baik dari data yang baru. Agar model dapat beradaptasi dengan data yang tidak sesuai pada saat pelatihan, model memanfaatkan metode monte carlo dropout untuk mencegah jaringan menjadi terlalu khusus pada contoh pelatihan tertentu dan meningkatkan generalisasi model. Dengan menggunakan arsitektur ResNet dan Monte Carlo dropout, model dapat mengurangi tingkat loss seiring proses pelatihan berlangsung, bahkan dengan proses pelatihan sebanyak 35 kali dengan jumlah batch sebanyak 32 tingkat akurasi dari model dapat mencapai 97% dan tingkat loss sebesar 6.5%.

Kata Kunci— *Residual Network*, *Monte Carlo Dropout*, *Deep Learning*, *Convolutional Neural Network*, Analisis Gambar Medis.

Abstract

Malaria is an acute febrile disease 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 nest in the liver. The similarity of its symptoms to common illnesses makes malaria difficult to recognize without microscopic examination of blood smears. However, the accuracy of microscopic examination depends on the quality of the smears and the expertise in classifying and counting infected and uninfected cells. Such examinations can be challenging for large-scale diagnosis and may result in poor quality. To overcome these limitations, a deep learning method called Residual Network can be used.

Residual network is an architecture of Convolutional Neural Network that allows the network to skip or bypass certain layers. Skip connections enable more efficient gradient flow during training and allow the network to learn better representations of new data. To adapt the model to data variations during training, the model utilizes the Monte Carlo dropout method to prevent the network from becoming overly specialized to specific training examples and improve the generalization of the model. By using the ResNet architecture and Monte Carlo dropout, the model can reduce the loss rate as the training process progresses. Even with 35 training iterations and a batch size of 32, the model can achieve an accuracy level of 97% and a loss rate of 6.5%.

Keywords— *Residual Network, Monte Carlo Dropout, Deep Learning, Convolutional Neural Network, Medical image analysis*

1. PENDAHULUAN

Malaria adalah penyakit demam akut yang disebabkan oleh parasit Plasmodium, yang ditularkan ke manusia melalui gigitan nyamuk *Anopheles* betina. Ketika nyamuk menggigit manusia, parasit akan memasuki tubuh manusia dan bersarang di hati[1]. Gejala pertama pada penderita malaria biasanya muncul 10-15 hari setelah gigitan nyamuk, dan gejala yang dialami ialah demam, sakit kepala dan menggigil[2] karena gejala yang dialami merupakan penyakit umum, hal itu menyebabkan gejala-gejala yang dialami sulit dikenali sebagai malaria. Pemeriksaan mikroskop dengan memeriksa hapusan darah pasien telah menjadi baku emas untuk diagnosa malaria, karena metodenya yang cepat dan murah[3]. Namun, akurasi pemeriksaan mikroskop tergantung pada kualitas hapusan, keahlian dalam mengklasifikasikan dan menghitung sel yang diparasit dan tidak terinfeksi[1]. pada jurnal yang ditulis oleh Afrizal Zein, ia menggunakan Bahasa python dengan model ResNet dengan metode yang hampir sama yaitu membagi data menjadi 3 jenis berupa train, test, dan valid tetapi dalam proses training data, peneliti menggunakan library keras yang tersedia di python dengan jumlah iterasi sebanyak 50 kali menghasilkan tingkat akurasi sebesar 96,5% pada training, 96,78% pada validation, dan 97% pada testing[4]. Penelitian ini akan menggunakan Python dan mengacu pada penelitian sebelumnya dengan dataset hapusan darah tipis yang terinfeksi dan tidak terinfeksi dengan masing masing sebanyak 13.780 citra, setelah itu pada proses training akan diterapkan metode monte carlo dropout dengan rate 45% dan menggunakan iterasi 35 kali untuk melihat Apakah ada pengaruh dalam hal peningkatan akurasi dari penelitian sebelumnya.

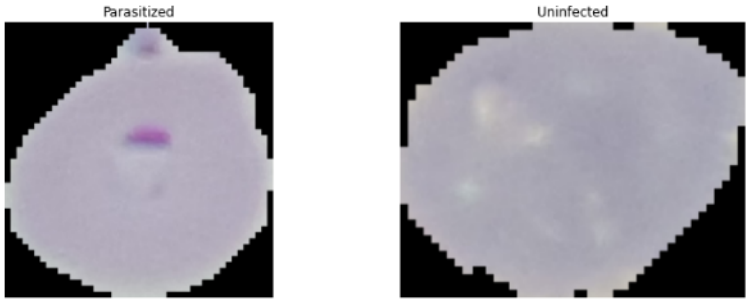
2. METODE PENELITIAN

A. Pengumpulan Data

Pada penelitian ini dilakukan beberapa metode pengumpulan data untuk mendapatkan data yang dibutuhkan. Metode yang digunakan adalah melakukan Studi Pustaka dengan mencari dataset yang digunakan oleh peneliti sebelumnya, dimana peneliti tersebut mendapatkan Dataset dari Chittagong Medical College Hospital di Bangladesh, di mana gambar apusan darah tipis dengan pewarnaan Giemsa difoto dari P. falciparum yang terinfeksi., dataset berupa hasil mikroskopik hapusan darah tipis dari pasien pengidap malaria dan pasien yang sehat dengan total jumlah dataset 13.780 pada masing masing kategori citra mikroskopik hapusan darah tipis.

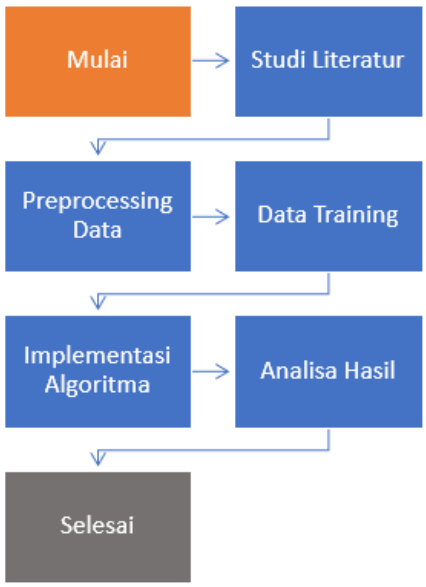
Data yang diperoleh dari Kaggle sudah dilakukan pemisahan antara dua jenis hapusan darah, yaitu antara yang terinfeksi dan tidak terinfeksi dipisah pada folder masing-masing, sehingga tahap yang perlu dijalankan adalah membuat program dapat membaca dua kelompok dataset hapusan darah yang terinfeksi dan yang tidak terinfeksi, data yang memiliki infeksi diberi label

parasitized dan yang tidak terinfeksi diberi label *uninfected*.



Gambar 1 Sampel data yang terinfeksi pada sebelah kiri dan yang tak terinfeksi pada sebelah kanan

Adapun tahapan penelitian yang akan dilakukan yang digambarkan dalam sebuah kerangka penelitian dibawah ini.



Gambar 2 Tahapan Penelitian

B. *Pre-Processing*

Tahap *pre-processing* dengan membagi dataset menjaditiga bagian yaitu *training, validation,* dan *testing*, sedangkan rasio dataset dibagi menjadi 70% untuk *training*, 20% untuk *validation,* dan 10% untuk *testing* menggunakan python dengan kode pada gambar 3.

```
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()
```

Gambar 3 Kode program untuk *split* dataset

Fungsi *load_train_images* memiliki tujuan untuk memuat gambar-gambar pelatihan dari setiap kelas dan menyimpannya ke folder tujuan yang sesuai. Fungsi ini menerima parameter opsional *n* yang mengontrol jumlah gambar yang akan dimuat untuk setiap kelas, dan data yang dibutuhkan untuk *training* berjumlah 9646 dari setiap label. Algoritma dari kode ini dipakai berulang ulang untuk membagi data pada *validation* dengan jumlah 2756 dan *testing* dengan jumlah 1378 pada masing masing label.

C. Normalisasi dan Augmentasi

Normalisasi digunakan untuk meningkatkan stabilitas numerik, mempercepat konvergensi algoritma, menghilangkan bias atau perbedaan skala, serta memudahkan interpretasi dan komparasi data. Dalam konteks pengolahan citra, normalisasi juga membantu mengatasi perbedaan intensitas cahaya, kontras, atau skala warna antara gambar-gambar yang berbeda.

Sedangkan augmentasi data adalah teknik yang digunakan untuk memperluas jumlah data pelatihan dengan menghasilkan variasi baru dari data yang ada. Ini dilakukan dengan menerapkan transformasi atau manipulasi sederhana pada gambar-gambar dalam dataset. Tujuan dari augmentasi data adalah untuk meningkatkan kemampuan generalisasi model dan mengurangi overfitting.

Pada tahap ini ukuran citra diubah menjadi 64x64 untuk mempercepat proses *training* dengan ukuran citra yang tidak terlalu besar namun masih sesuai dengan citra aslinya sedangkan untuk konfigurasi augmentasi dapat dilihat pada tabel 1.

TABLE I. KONFIGURASI AUGMENTASI

Parameter	Konfigurasi
Horizontal_Flip	True
Rotation_range	30
Zoom_range	0.2

D. *Residual Network*

Residual Network atau ResNet merupakan arsitektur untuk algoritma CNN yang dikembangkan oleh peniliti dari Microsoft Research. Arsitektur ini merupakan suatu metode untuk menambahkan potongan layer (*residual*), atau skip, connection yang secara langsung menambahkan output setiap layer ke *output layer* berikutnya. Metode ini dibentuk untuk mengatasi salah satu masalah pada deep learning yaitu exploding dan vanishing gradient[5].

Residual Network (ResNet) menggunakan blok-blok residu atau *residual blocks* yang memungkinkan jaringan untuk melompati atau *skip* beberapa lapisan, *skip connection* memungkinkan aliran gradien yang lebih efisien selama pelatihan dan memungkinkan jaringan untuk belajar representasi yang lebih baik dari data yang baru. Kode program untuk membangun arsitektur ResNet bisa dilihat pada Gambar 14.

```
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
```

Gambar 4 Kode untuk unit *ResNet*

Fungsi *resnet_unit* adalah untuk membangun blok unit dalam arsitektur ResNet. Blok unit ini terdiri dari beberapa lapisan konvolusi, normalisasi batch, dan skip connection. Fungsi ini mengambil input dari tensor input dari layer sebelumnya, jumlah *filters* untuk lapisan konvolusi, dan beberapa parameter lain seperti jumlah lapisan konvolusi (*n_conv*), ukuran kernel (*kernel_size*), dan langkah (*strides*). Fungsi ini mengembalikan *output* yang merupakan hasil dari blok unit ResNet.

Berikut adalah alur algoritma untuk resnet_unit:

1. Menginisialisasi variabel x dengan nilai inputs.
2. Melakukan iterasi sebanyak *n_conv* kali.

a. Menerapkan lapisan konvolusi dengan jumlah filter filters, ukuran kernel, langkah, fungsi aktivasi ReLU, dan padding *same* ke x.

b. Melakukan normalisasi batch pada x menggunakan *BatchNormalization*.

3. Menggabungkan keluaran x dengan input awal inputs menggunakan *Concatenate*.
4. Menerapkan fungsi aktivasi ELU pada hasil gabungan sebelumnya.
5. Mengembalikan hasil akhir out.

Setelah membentuk unit untuk ResNet, fungsi ini akan dimasukkan kedalam model yang akan dibentuk sebagai arsitektur ResNet yang akan digunakan sebagai model.

E. *Monte Carlo Dropout*

Dropout pada umumnya adalah metode yang bertujuan untuk mencegah overfitting pada neural networks, metode ini bekerja dengan menghilangkan neuron berdasarkan rate yang sudah ditentukan pada saat pelatihan, tetapi pada saat pelatihan selesai semua neuron akan tetap aktif[6].

Monte Carlo Dropout adalah metode yang diusulkan oleh Gal dan Ghahramani. Mereka menginterpretasikan dropout sebagai metode pengambilan sampel yang setara dengan perkiraan variasi dari deep Gaussian process. Metode dropout ini adalah metode yang dapat menghasilkan estimasi ketidakpastian model dengan cara yang sejalan. Monte Carlo Dropout adalah teknik yang digunakan untuk mengurangi overfitting pada saat pelatihan model. Pada Monte Carlo Dropout, selama fase pengujian, dropout diterapkan dengan menggunakan dropout stokastik, yaitu melakukan dropout pada beberapa unit secara acak dengan probabilitas tetap, dan melakukannya secara berulang. Hasil prediksi diperoleh dengan melakukan inferensi ulang beberapa kali dan menggabungkan hasil-hasil tersebut. Teknik ini menghasilkan perkiraan yang

lebih stabil dan memperhitungkan ketidakpastian model. Ini mencegah jaringan menjadi terlalu khusus pada contoh pelatihan tertentu dan meningkatkan generalisasi model.

Monte Carlo Dropout telah menemukan banyak aplikasi dalam praktik, termasuk dalam prediksi deret waktu dan pencitraan medis.[6]

Pada penelitian ini pembuatan fungsi monte carlo dropout untuk arsitektur ResNet adalah sebagai berikut

```
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)
```

Gambar 5 Kode *Monte Carlo Dropout*

Berikut alur program dari algoritma Monte Carlo Dropout diatas:

a) Inisialisasi model: Model *MCDropoutModel* dibuat dengan menginisialisasi objek dropout layer dengan tingkat dropout yang ditentukan.

b) Mode pelatihan: Saat model dipanggil dengan parameter *training=True*, mode pelatihan diaktifkan.

c) Dropout aktif: Pada mode pelatihan, dropout layer diterapkan pada input dengan menggunakan metode *dropout_layer(inputs, training=True)*. Hal ini mengaktifkan dropout, di mana setiap elemen input memiliki peluang *rate* untuk di-dropout secara acak.

d) Mode inferensi: Saat model dipanggil dengan parameter *training=False*, mode inferensi atau prediksi diaktifkan.

e) Dropout dinonaktifkan: Pada mode inferensi, dropout tidak diterapkan pada input, sehingga metode *dropout_layer(inputs, training=False)* akan mengembalikan input tanpa perubahan.

Saat mode pelatihan, dropout diterapkan pada input untuk membantu dalam regularisasi dan mengurangi overfitting. Saat mode inferensi, dropout dinonaktifkan untuk mendapatkan prediksi yang stabil dan mengestimasi ketidakpastian prediksi dengan menggunakan metode Monte Carlo Dropout.

f. Tinjauan Pustaka

Berikut tabel penelitian yang berkaitan dengan penelitian yang menggunakan model ResNet

1. Implementation of Convolutional Neural Networks for Batik Image Dataset, 2022. Model CNN dikonstruksi dengan menggunakan lapisan konvolusi dan pooling untuk mengekstrak fitur-fitur penting dari gambar batik, Model memiliki tingkat akurasi sebesar 95%. Selain itu, model juga berhasil mengatasi tantangan dalam mengklasifikasikan gambar batik dengan variasi skala, rotasi, dan distorsi.

2. Vehicle detection using improved region convolution neural network for accident prevention in smart roads, 2022. Metode yang diusulkan menggabungkan penggunaan Region Proposal Network (RPN) dengan Convolutional Neural Network (CNN) untuk meningkatkan akurasi deteksi kenderaan, Penulis melakukan evaluasi pada dataset kendaraan dan menemukan bahwa model yang diusulkan berhasil mencapai tingkat akurasi deteksi sebesar 98,68%, dengan kecepatan pemrosesan mencapai 30% per detik.
3. Li-ion battery degradation modes diagnosis via Convolutional Neural Networks, 2022. Penulis mengusulkan pendekatan yang menggunakan dua arsitektur CNN yang berbeda untuk mengekstraksi fitur dari data degradasi baterai, dan kemudian menggunakan model regresi logistik untuk memprediksi mode degradasi yang mungkin terjadi, Dalam cross-validation, model CNN yang menggunakan arsitektur ResNet-50 memiliki kinerja yang lebih baik dalam memprediksi mode degradasi dengan akurasi 95,1%, dibandingkan dengan model CNN yang menggunakan arsitektur VGG-16 dengan akurasi 91,6%..
4. COVID-19 infection prediction from CT scan images of lungs using Iterative Convolution Neural Network model, 2022. Metode ini dievaluasi pada dataset gambar CT yang dikumpulkan dari pusat medis yang berbeda dan memperoleh hasil yang mengesankan dalam memprediksi infeksi COVID-19. Proses training juga menggunakan 2 model dengan CNN1 memiliki 2 *hidden layer* dan CNN2 memiliki 3 hidden layer, Model yang dirancang dengan 100 set data latih pada kedua CNN1 dan CNN2, telah menghasilkan akurasi yang lebih tinggi dalam klasifikasi COVID dibandingkan dengan model lainnya. Akurasi klasifikasi terbaik sebesar 89% dicapai dari model CNN2 pada iterasi ke-5.
5. Deep Convolution Neural Network sharing for the multi-label images classification, 2022. membagi data gambar menjadi serangkaian sub-gambar dan kemudian melatih setiap DCNN (Deep Convolutional Neural Network) pada subset dari sub-gambar Setiap DCNN dilatih untuk mengidentifikasi subset label tertentu, dan keluaran dari semua DCNN kemudian digabungkan untuk membuat keputusan klasifikasi akhir., Jaringan dengan multi-fitur mendapatkan skor klasifikasi tertinggi dibandingkan jaringan syaraf dalam dengan 83,31% skor f1 untuk dataset Amazon rainforest. Nilai skor f1 adalah 88,81% untuk dataset Pascal VOC 2007, 87,71% untuk Nuswide, dan 88,64% untuk Pascal VOC 2012.

3. HASIL DAN PEMBAHASAN

Arsitektur pada penelitian ini terdiri dari beberapa jenis layer yang terhubung satu sama lain. Input dari model ini berupa citra dengan ukuran 64x64 piksel dan 3 channel warna, layer pertama dalam model adalah *Convolutional* 2 dimensi dengan 8 filter dengan ukuran 3x3 piksel. Setelahnya, dilakukan *batch normalization* untuk normalisasi nilai dalam layer. Proses ini diulangi sebanyak 4 kali dengan jumlah filter yang sama pada setiap langkahnya, setelah itu, hasil dari layer *Convolutional* dan *batch normalization* digabungkan dengan input awal menggunakan operasi *concatenate*. Hasilnya kemudian dijalankan melalui fungsi aktivasi ELU dan dilakukan *average pooling* dengan ukuran 2x2 piksel.

Proses ini dilanjutkan dengan layer *Convolutional*, *batch normalization, concatenation*, dan *average pooling* yang dilakukan beberapa kali dengan peningkatan jumlah filter pada setiap langkahnya. Pada tahap selanjutnya, hasil dari proses sebelumnya dijalankan melalui beberapa layer *convolutional* dan *batch normalization* dengan jumlah filter yang semakin bertambah. Setelahnya, dilakukan *concatenation* dan *average pooling*. Langkah-langkah di atas diulangi kembali dengan jumlah filter yang lebih besar hingga didapatkan hasil akhir dengan ukuran 2x2 piksel dan 507 *channel*. Hasil tersebut kemudian diflatten menjadi vektor 1 dimensi dan dihubungkan dengan layer *dense* dengan 4096 unit. Terdapat juga layer mc_dropout_model

yang merupakan modifikasi dari layer *dense* yang menggunakan metode dropout untuk mengurangi *overfitting*, akhirnya, terdapat layer Dense terakhir dengan 4096 unit yang menjadi output dari model ini.

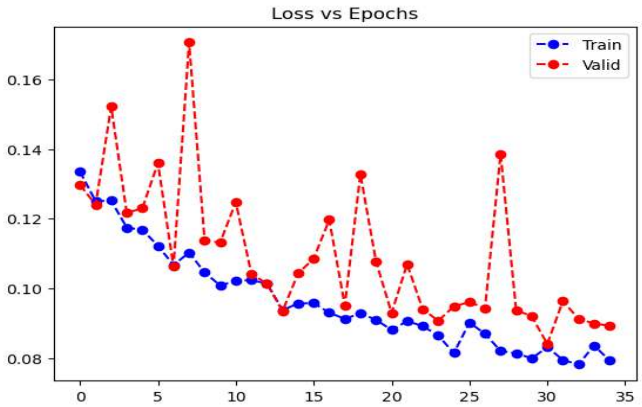
Pada pelatihan kali ini *batch* yang digunakan sebanyak 32 dengan jumlah iterasi sebanyak 35 kali untuk mengetahui seberapa banyak peningkatan pada *loss* dan akurasi dibandingkan validasi. Batch size adalah jumlah sampel data yang digunakan dalam satu iterasi pelatihan pada proses pelatihan model.

Hasil dari iterasi pelatihan model dapat dilihat pada Gambar 6

```
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
```

Gambar 6 Proses pelatihan model

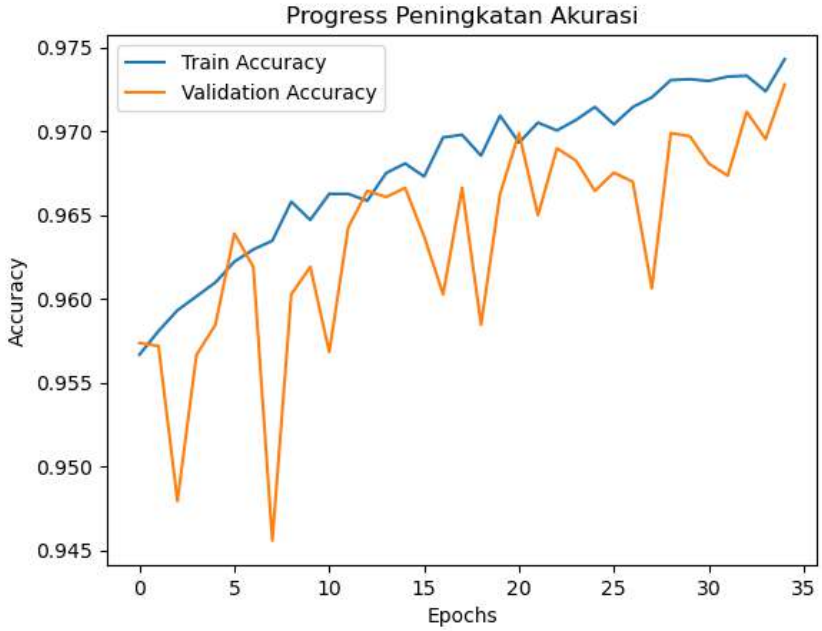
Setelah hasil pelatihan didapatkan, langkah selanjutnya adalah visualisasi data untuk melihat perkembangan model selama proses training dilakukan.



Gambar 7 Plot nilai *loss* selama pelatihan

Nilai *loss* pelatihan mengalami penurunan seiring dengan peningkatan jumlah *epoch*,

menunjukkan bahwa model mampu mengurangi kesalahan prediksi. Meskipun terdapat fluktuasi kecil, *loss* validasi secara keseluruhan cenderung stabil dan menunjukkan bahwa model tidak overfitting.



Gambar 8 Nilai Akurasi pada pelatihan

Akurasi pelatihan meningkat seiring dengan jumlah epoch, mencapai sekitar 97% pada akhir pelatihan. Hal ini menunjukkan bahwa model berhasil mempelajari pola yang ada dalam data pelatihan. Akurasi validasi juga meningkat dan mencapai sekitar 97%, menunjukkan bahwa model dapat melakukan prediksi dengan baik pada data yang belum pernah dilihat sebelumnya.

4. KESIMPULAN

Setelah model dibentuk dan dievaluasi, hasil penelitian model *ResNet* dengan *monte carlo dropout,* didapatkan beberapa kesimpulan.

1. Akurasi pelatihan meningkat seiring dengan jumlah epoch, mencapai sekitar 97% pada akhir pelatihan. Hal ini menunjukkan bahwa model berhasil mempelajari pola yang ada dalam data pelatihan.
2. L*oss* pelatihan mengalami penurunan seiring dengan peningkatan jumlah *epoch*, menunjukkan bahwa model mampu mengurangi kesalahan prediksi.
3. Akurasi validasi juga meningkat dan mencapai sekitar 97%, menunjukkan bahwa model dapat melakukan prediksi dengan baik pada data yang belum pernah dilihat sebelumnya.
4. Terdapat fluktuasi kecil, *loss* validasi secara keseluruhan cenderung stabil dan menunjukkan bahwa model tidak overfitting.
5. Model ini dapat digunakan untuk melakukan prediksi malaria pada citra hapusan darah tipis.

5. SARAN

Berdasarkan hasil pelatihan model, terdapat beberapa saran yang dapat diberikan berupa:

1. Meskipun akurasi model pada data pelatihan dan validasi sudah cukup tinggi, namun ada kemungkinan masih terdapat ruang untuk meningkatkan performa model. Salah satu pendekatan yang dapat dilakukan adalah dengan melakukan pengaturan pada parameter-parameter model, seperti *optimizer*, *learning rate*, atau arsitektur jaringan. Dengan melakukan eksperimen lebih lanjut, mungkin akan ditemukan konfigurasi yang lebih optimal untuk meningkatkan akurasi dan mengurangi loss model.
2. Dataset yang digunakan pada penelitian ini bukan dataset yang didapatkan secara langsung melalui rumah sakit, sehingga model perlu dilakukan pelatihan menggunakan dataset langsung yang dapat diperoleh di rumah sakit yang masih memiliki banyak kasus penyakit malaria.

DAFTAR PUSTAKA

[1] W. Kusuma, A. A. W. Lestari, S. Herawati, W. Putu, and S. Yasa, "Pemeriksaan Mikroskop Dan Tes Diagnostik Cepat Dalam Menegakkan Diagnosis Malaria", 2014. doi: 10.24843/eum.

[2] World Health Organization, "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, "Perbandingan Efektifitas Rapid Diagnostic Test (RDT) dengan Pemeriksaan Mikroskop pada Penderita Malaria Klinis," Manado, Jul. 2018.

[4] A. Zein, "Pendeteksian Penyakit Malaria Menggunakan Medical Images Analisis Dengan Deep Learning Python 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 - Gejala, penyebab dan mengobati - 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.