Deteksi Penyakit Mata Pada Citra Fundus Menggunakan Convolutional Neural Network (CNN)

Ocular Disease Detection on Fundus Images Using Convolutional Neural Network (CNN)

Rarasmaya Indraswari[1], Wiwiet Herulambang[2], Rika Rokhana[3]
[1]Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember (ITS)
[2]Program Studi Teknik Informatika, Universitas Bhayangkara, Surabaya
[3]Departemen Teknik Elektro, Politeknik Elektronika Negeri Surabaya (PENS)
E-mail: [1]raras@its.ac.id, [2]herulambang@ubhara.ac.id, [3]rika@pens.ac.id

Abstrak

Pada tahun 2020, terdapat 1,1 milyar orang yang mengalami kehilangan penglihatan di seluruh dunia. Jumlah ini diproyeksikan akan terus bertambah hingga mencapai 1,76 milyar orang pada tahun 2050. Penyebab utama kebutaan untuk anak-anak dan remaja adalah penyakit mata, yang dapat dicegah apabila dilakukan deteksi dan penanganan lebih dini. Oleh sebab itu, pada penelitian ini diusulkan metode berbasis *Convolutional Neural Network* (CNN) untuk mendeteksi penyakit mata pada citra fundus. Metode yang diusulkan menggunakan metode *transfer learning* dengan arsitektur jaringan MobileNetV2 sebagai *base model*. Arsitektur *head model* yang diusulkan, yang terdiri dari lapisan *global average pooling* dan diikuti oleh 2 lapisan *fully-connected*, mampu memberikan akurasi yang paling tinggi dan efisiensi paling baik dibandingkan dengan arsitektur *head model* lainnya. Eksperimen pada dataset citra fundus yang terdiri dari 601 citra dengan berbagai macam penyakit mata menunjukkan bahwa metode yang diusulkan mampu memberikan performa yang baik dengan nilai akurasi sebesar 72%, *precision* sebesar 72%, *recall* sebesar 72%, dan F1-*score* sebesar 72%. Hasil eksperimen menunjukkan bahwa metode yang diusulkan dapat memberikan akurasi yang lebih tinggi dan lebih efisien dibandingkan dengan menggunakan arsitektur CNN lainnya, seperti ResNet50V2, InceptionV3, InceptionResNetV2, VGG16, dan VGG19.

Kata kunci: penyakit mata, *convolutional neural network* (CNN), MobileNetV2, citra fundus, klasifikasi citra

Abstract

In 2020, there was 1.1 billion people who have lost their sight worldwide. This number will continue to grow until it reaches 1.76 billion people by 2050. The main cause of blindness for children and adolescents is ocular disease, which can be done if detected and treated early. Therefore, in this study, we proposed a method based on convolutional neural network (CNN) to detect ocular disease in fundus images. The proposed method uses transfer learning method with MobileNetV2 network architecture as the base model. The proposed head model architecture, which consists of a global average pooling layer followed by 2 fully-connected layers, is able to provide the highest accuracy and best efficiency compared to other head model architectures. Experiments on a fundus image dataset consisting of 601 images with various ocular diseases show that the proposed method is able to provide good performance with an accuracy of 72%, precision of 72%, recall of 72%, and F1-score of 72%. The experimental results show that the proposed method provides higher accuracy and efficiency compared to other CNN architectures, such as ResNet50V2, InceptionV3, InceptionResNetV2, VGG16, and VGG19.

Keywords: ocular disease, convolutional neural network *(CNN), MobileNetV2, fundus image, image classification*

1. PENDAHULUAN

Mata (*oculus*) merupakan salah satu dari organ vital tubuh yang sangat penting. Mata termasuk salah satu dari 5 panca indra manusia. Manusia dapat memperoleh informasi sebanyak 80% hanya dengan melihat [1]. Menjaga mata tetap sehat dengan mengkonsumsi makanan bergizi seimbang dan olahraga yang cukup serta mengurangi paparan *ultraviolet* secara langsung dari perangkat elektronik adalah hal yang perlu dan harus dilakukan untuk mencegah atau mengurangi dampak negatif yang disebabkan oleh aktivitas berlebih pada mata. Dampak negatif yang diterima oleh mata akan mengakibatkan ketidaknormalan pada mata, seperti *cataract, glaucoma,* dan *retina disease.* Apabila tidak segera ditangani, penyakit mata dapat menyebabkan kebutaan. Berdasarkan laporan dari Vision Atlas pada tahun 2020, terdapat sebanyak 1,1 milyar orang yang kehilangan penglihatan di seluruh dunia [2]. Dari jumlah tersebut, sebanyak 90 juta orang di antaranya adalah anak-anak dan remaja, di mana 2,1 juta orang di antaranya mengalami kebutaan [2]. Jumlah ini diproyeksikan akan terus bertambah hingga mencapai 1,76 milyar orang di seluruh dunia yang mengalami kehilangan penglihatan pada tahun 2050 [2]. Penyebab utama kebutaan untuk anak-anak dan remaja adalah penyakit mata [3]. Diperkirakan bahwa sebanyak 40% dari kebutaan yang terjadi pada anak-anak dan remaja akibat penyakit mata dapat dicegah apabila penyakit tersebut dapat dideteksi dan ditangani lebih dini [3].

Citra fundus adalah salah satu jenis citra yang banyak digunakan untuk mendeteksi ketidaknormalan pada mata, seperti penyakit *diabetic retinopathy*, *glaucoma*, katarak, hipertensi, *myopia*, dll [4]. Citra ini merupakan representasi dua dimensi (2D) dari jaringan retina tiga dimensi (3D) yang diambil menggunakan mikroskop khusus berdaya rendah. Citra fundus bersifat non-invasif dan memiliki biaya yang cukup murah sehingga efektif digunakan untuk melakukan deteksi penyakit mata. Deteksi penyakit mata biasanya dilakukan berdasarkan analisis manual oleh dokter (*expert*) pada citra fundus sehingga hasil analisnya sangat tergantung pada pengalaman dari dokter tersebut. Perbedaan hasil analisis antar satu dokter dengan dokter lainnya sangat mungkin terjadi (*inter-observer error*). Oleh sebab itu, telah dikembangkan beberapa metode visi komputer (*computer-aided diagnosis* / CAD) untuk membantu dokter dalam melakukan analisis penyakit mata.

Pada umumnya, sistem CAD menggunakan metode konvensional yang secara otomatis mensegmentasi objek-objek pada citra fundus yang diduga sebagai tanda ketidaknormalan pada retina, mengekstrak fitur-fitur dari objek tersebut, dan menggunakannya untuk menentukan apakah benar objek tersebut menunjukkan ketidaknormalan pada area retina. Metode ini memiliki tingkat kesulitan yang tinggi pada tahap segmentasi objek secara otomatis, di mana objek satu dengan yang lain memiliki bentuk, warna, dan ukuran yang berbeda. Oleh sebab itu, banyak penelitian yang mengusulkan metode segmentasi, baik segmentasi otomatis maupun semi-otomatis, untuk memisahkan objek abnormal dengan struktur lainnya pada citra fundus. Namun, karena perbedaan tanda-tanda ketidaknormalan satu dengan yang lain, seperti antara *glaucoma* dan *retinal disease*, maka biasanya metode deteksi secara konvensional ini hanya spesifik dapat digunakan untuk satu jenis penyakit saja. Untuk mendeteksi jenis penyakit lainnya, diperlukan algoritma lain untuk melakukan segmentasi objek abnormal penanda penyakit tersebut dan mengekstrak fitur-fiturnya.

Di lain pihak, saat ini telah berkembang metode *deep learning* yang merupakan perkembangan dari metode *machine learning artificial neural network* (ANN). ANN dikembangkan berdasar struktur *neuron* pada otak yang saling berhubungan untuk melakukan pengambilan keputusan berdasarkan *input* yang diberikan. Pada klasifikasi citra menggunakan metodologi konvensional, metode *machine learning* seperti ANN biasanya digunakan untuk mengklasifikasi kelas berdasar data numerik yang telah diekstrak dari objek pada citra. Oleh sebab itu, hasil klasifikasi dengan metode konvensional sangat tergantung pada hasil segmentasi objek dan ekstraksi fitur objek tersebut. Metode *deep learning* untuk klasifikasi citra, seperti *Convolutional Neural Network* (CNN), menghilangkan dependensi terhadap proses segmentasi dan ekstraksi fitur, di mana seluruh citra menjadi input dari *deep learning network* dan

kemudian sistem yang akan langsung mempelajari fitur mana saja dari citra yang penting untuk menentukan kelas dari citra.

Secara umum, klasifikasi citra menggunakan CNN maupun berbagai pengembangannya memberikan hasil akurasi yang tinggi apabila didukung dengan jumlah data *training* yang cukup. Gulshan, et al. (2016) mengembangkan algoritma *deep learning* untuk mendeteksi penyakit mata *diabetic retinopathy*. Hasil penelitian ini menunjukkan bahwa algoritma *deep learning* dapat digunakan untuk melakukan klasifikasi penyakit *diabetic retinopathy* dengan akurasi yang tinggi [5]. Diaz-Pinto, et al. (2019) melakukan deteksi *glaucoma* pada 1707 citra fundus menggunakan 5 arsitektur CNN yang berbeda (VGG16, VGG19, InceptionV3, ResNet50, dan Xception). Hasil terbaik diperoleh oleh arsitektur Xception dengan rata-rata nilai AUC sebesar 96%, *specificity* sebesar 85.8%, dan *sensitivity* sebesar 93.5% [6]. Li, et al. (2019) mengusulkan metode *attention-based* CNN untuk mendeteksi *glaucoma*. Hasil eksperimen terhadap 5.824 citra fundus memberikan nilai akurasi deteksi sebesar 95.3%, *sensitivity* sebesar 95.4%, *specificity* sebesar 95.2%, dan AUC sebesar 97.5% [7]. Bajwa, et al. (2019) mengusulkan metode berbasis *Regions with Convolutional Neural Network* (RCNN) untuk melakukan segmentasi dan klasifikasi *glaucoma.* Dari penelitian ini dapat disimpulkan bahwa metode *deep learning* dapat melakukan deteksi dan lokalisasi objek dengan *robust*, akurat, dan otomatis apabila dilatih pada dataset beranotasi berukuran besar [8]. Gangwar & Ravi (2020) menggunakan arsitektur InceptionResNetV2 untuk melakukan deteksi penyakit *diabetic retinopathy* pada citra fundus dengan akurasi sebesar 72.33% untuk dataset MESSIDOR-1 dan 82.18% untuk dataset Kaggle [9]. Joshi, et al. (2020) menggunakan arsitektur YOLO-v3 untuk mendeteksi penyakit *glaucoma* pada 2000 citra fundus dengan akurasi sebesar 93.7%, *sensitivity* sebesar 89.1%, dan *specificity* sebesar 95.8% [10].

Penelitian sebelumnya umumnya menggunakan metode berbasis CNN untuk mendeteksi satu jenis penyakit mata. Namun, deteksi ketidaknormalan pada mata, yang bisa disebabkan oleh berbagai macam penyakit, masih belum banyak dilakukan. Oleh sebab itu, pada penelitian ini diusulkan untuk melakukan deteksi penyakit mata pada citra fundus menggunakan metode CNN. Pada penelitian ini dilakukan pengembangan arsitektur jaringan MobileNetV2 dan menggunakan metode *transfer learning* untuk melakukan deteksi ada tidaknya penyakit mata pada dataset citra fundus. Jaringan MobileNetV2 memiliki arsitektur yang ringan sehingga memungkinkan data diproses lebih cepat dibandingkan dengan arsitektur CNN lainnya seperti VGG, ResNet, dsb [11]. Pada penelitian ini diusulkan suatu arsitektur *head model* di atas jaringan MobileNetV2, yang terdiri dari sebuah lapisan *global pooling* yang diikuti dengan dua buah lapisan *fully-connected*. Arsitektur *head model* yang diusulkan ini dapat mempertahankan efisiensi jaringan karena tidak menggunakan terlalu banyak parameter untuk dilatih, sekaligus memberikan hasil akurasi yang paling baik berdasarkan hasil uji coba yang dilakukan. Metode yang diusulkan akan dilatih dan diujicobakan pada dataset yang terdiri dari beberapa jenis penyakit mata, yaitu *cataract*, *glaucoma, retinal disease*, yang digabungkan menjadi satu kelas, yaitu kelas "Abnormal". Dengan demikian, metode yang diusulkan mampu mendeteksi ketidaknormalan pada mata, tanpa terpaku pada satu jenis penyakit mata. Diharapkan bahwa metode yang diusulkan dapat memberikan hasil yang akurat sehingga bisa dimanfaatkan untuk *computer-aided diagnosis system* dan membantu deteksi dini penyakit mata untuk meningkatkan kualitas hidup masyarakat.

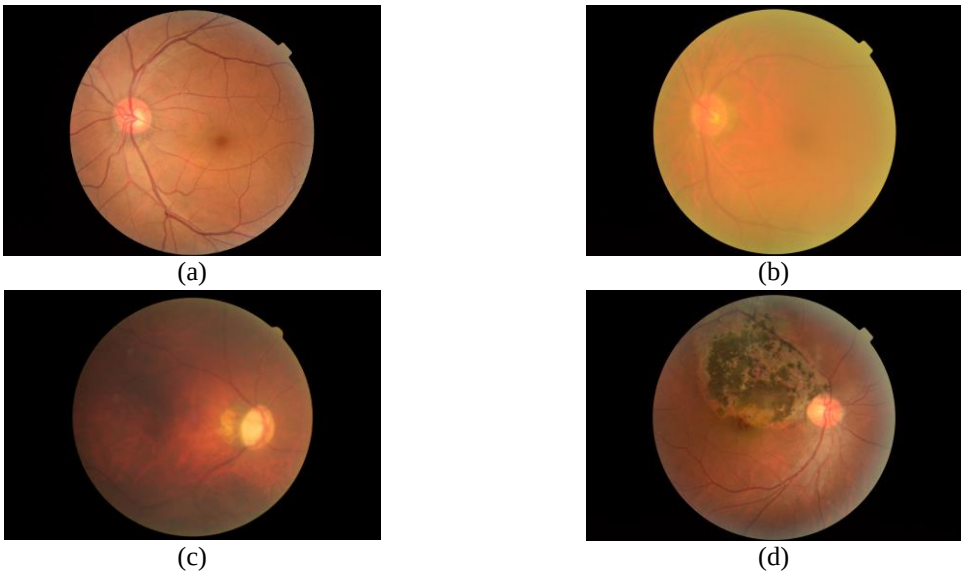
2. METODE PENELITIAN

2.1 Dataset

Dataset citra fundus merupakan data sekunder yang diambil dari Kaggle (https://www.kaggle.com/datasets/jr2ngb/cataractdataset) yang terdiri dari 601 buah citra fundus. Terdapat 4 kelas pada dataset tersebut, yaitu kelas "Normal" (300 citra), "Cataract" (100 citra), "Glaucoma" (101 citra), dan "Retina Disease" (100 citra). Pada penelitian ini, kelas pada dataset tersebut dikelompokkan menjadi 2 kelas, yaitu kelas "Normal" dan "Tidak Normal", di mana kelas "Tidak Normal" terdiri dari citra pada kelas "Cataract", "Glaucoma", dan "Retina

Disease", sehingga total terdapat 301 citra pada kelas "Tidak Normal". Contoh citra dari kelas "Normal" dan "Abnormal (*cataract, glaucoma, retinal disease*)" secara berturut-turut ditunjukkan pada Gambar 1.

Karena pada penelitian ini dataset dikelompokkan ke dalam dua kelas dengan jumlah data per kelas yang hampir sama, maka pada penelitian ini tidak terdapat permasalahan terkait ketidakseimbangan data pada dataset (*imbalanced dataset*) yang sangat mempengaruhi performa akurasi algoritma *deep learning* [12], [13]. Pada penelitian ini, dataset dibagi menjadi data *training* dan *testing* dengan perbandingan 7:3, sehingga total terdapat 420 data *training* dan 181 data *testing*. Dari 420 buah data *training*, 210 buah di antaranya merupakan data dari kelas "Normal" dan 210 buah di antaranya merupakan data dari kelas "Abnormal". Dari 181 buah data *testing*, 90 buah di antara merupakan data dari kelas "Normal" dan 91 buah di antaranya merupakan data dari kelas "Abnormal".



(a) (b) (c) (d)

Gambar 1. Contoh citra fundus dari kelas (a) normal, (b) *cataract*, (c) *glaucoma*, dan (d) *retinal disease*

2.2 Convolutional Neural Network (CNN)

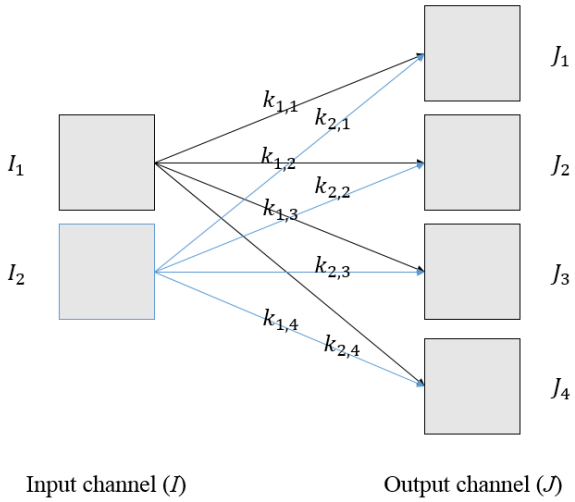
Metode *deep learning* semakin banyak digunakan dalam pengolahan citra karena dapat memberikan akurasi yang tinggi. Metode *deep learning* yang banyak digunakan untuk klasifikasi citra adalah CNN (*Convolutional Neural Network*). Secara umum, terdapat 3 jenis proses atau lapisan *(layer*) yang digunakan pada CNN, yaitu proses konvolusi, proses *pooling*, dan proses *fully-connected* (*dense layer*). Pada CNN, dilakukan serangkaian proses konvolusi 2D dengan ReLU (*rectified linear units*) [14] sebagai fungsi aktivasi di mana digunakan ukuran kernel konvolusi dengan ukuran tertentu (biasanya sebesar 3×3 piksel) dan untuk masing-masing proses konvolusi dihasilkan sebanyak beberapa buah *channel* atau *feature map*. Pada proses konvolusi, dilakukan *filtering* antara masing-masing *input channel* proses konvolusi dengan kernel-kernel tertentu. Hasil *filtering* pada setiap *input channel* menggunakan kernel-kernel tersebut kemudian dijumlahkan untuk menghasilkan sebuah *output channel* proses konvolusi. Ilustrasi untuk proses konvolusi ditunjukkan pada Gambar 2.

Apabila citra *input* proses konvolusi terdiri dari 2 buah *channel* (I_i; $i = \{1,2\}$) dan proses konvolusi yang dilakukan akan menghasilkan citra *output* dengan 4 buah *channel* (J_j; $j = \{1,2,3,4\}$), maka akan digunakan sebanyak total 8 buah kernel konvolusi $k_{i,j}$. Untuk menghasilkan output *channel* J_j, digunakan Persamaan 1 di mana $*$ merupakan simbol untuk proses konvolusi dan *N* merupakan jumlah *channel* pada citra *input*. Citra hasil proses konvolusi kemudian diaktivasi menggunakan suatu fungsi aktivasi, di mana saat ini fungsi aktivasi yang banyak digunakan adalah fungsi ReLU. Fungsi ReLU mengubah nilai *input x* dari hasil proses

konvolusi menjadi nilai output *f(x)* menggunakan Persamaan 2 [14]. Fungsi ReLU meningkatkan *sparsity* dari jaringan dan membuat proses *training* menjadi lebih cepat [12].

$$J_j = \sum_{i=1}^{N} I_i * k_{i,j} \qquad (1)$$

$$f(x) = \max(0, x) \qquad (2)$$



Gambar 2. Ilustrasi layer konvolusi

Pada CNN juga terdapat proses *pooling* dengan ukuran *window* tertentu (biasanya 2×2 piksel) untuk melakukan kompresi citra dan mengambil fitur-fitur yang penting. Proses *pooling* membagi citra *input* proses tersebut menjadi beberapa bagian dengan ukuran tertentu (*window*). Dari setiap bagian tersebut hanya akan diambil 1 piksel yang memiliki nilai yang merepresentasikan nilai statistik tertentu, tergantung dari jenis *pooling* yang digunakan. Apabila yang digunakan adalah proses *max pooling*, maka yang diambil adalah nilai tertinggi yang terdapat pada *window*. Apabila yang digunakan adalah proses *average pooling*, maka yang diambil adalah nilai rata-rata dari *window* tersebut, dan sebagainya. Dengan adanya proses *pooling*, maka ukuran citra akan menjadi lebih kecil sehingga komputasi yang dibutuhkan lebih rendah. Selain itu, proses ini juga membuat citra menjadi *rotation invariant*, karena lokasi dari piksel yang terpilih dalam *window* awal tidak diambil.

Proses terakhir yang terdapat pada arsitektur CNN adalah lapisan *fully-connected* atau *dense layer*. Secara umum, proses yang terdapat pada lapisan *fully-connected* sama persis dengan proses yang terdapat pada metode *artificial neural network* (ANN) di mana lapisan pada ANN menerima data numerik dan mengalikannya dengan nilai bobot untuk menghasilkan nilai *output* dari *nodes* pada lapisan tersebut. *Input* dari lapisan *fully-connected* pada CNN ini akan dijadikan sebagai matriks 1 dimensi (agar sama dengan *input* pada ANN) dan kemudian akan dilakukan pemetaan dari *input* ke sejumlah *node* yang terdapat pada lapisan *fully-connected*. Selanjutnya *output* dari lapisan *fully-connected* akan dipetakan ke beberapa *node* yang jumlahnya sesuai dengan jumlah kelas pada data. *Output* dari *node* kelas tadi adalah besar probabilitas data *input* merupakan anggota kelas tertentu. *Node* dengan probabilitas tertinggi akan menjadi kelas dari data *input*. Berikutnya dilakukan perhitungan *error* antara hasil klasifikasi dari *network* dan label/target dari data menggunakan *cost function* tertentu [15]. Nilai *error* ini kemudian digunakan untuk memperbarui nilai-nilai filter konvolusi pada network. Proses ini disebut sebagai tahap *back-propagation* dari proses *training* jaringan. Sedangkan proses untuk mendapatkan *output* sistem dari data *input* yang diberikan disebut sebagai tahap *feed forward* [15].

Terdapat banyak pengembangan dari arsitektur dasar CNN. Perbedaan dari bermacam-macam arsitektur tersebut antara lain terletak pada jumlah lapisan pada jaringan, variasi dari

peletakan lapisan konvolusi, *pooling*, dan *fully-connected*, serta adanya beberapa proses lain yang dikembangkan untuk meningkatkan performa dari jaringan tersebut. Contoh dari arsitektur CNN yang banyak digunakan antara lain adalah VGG [16], ResNet [17], Inception [18], InceptionResNet [19], dan MobileNet [11], [20].

2.3 Transfer Learning

Secara umum, metode *deep learning* memiliki kelemahan yaitu membutuhkan dataset berukuran besar agar dapat memberikan performa yang baik. Hal ini karena apabila menggunakan dataset berukuran kecil, metode *deep learning* kesulitan untuk mendapatkan informasi yang cukup mengenai variasi data dan tidak dapat mempelajari fitur yang membedakan kelas-kelas pada data. Pada metode *deep learning*, lapisan-lapisan awal pada jaringan bertugas untuk mempelajari struktur dan fitur umum pada data, di mana struktur dan fitur yang dipelajari ini semakin mengerucut seiring dengan kedalaman lapisan jaringan. Akhirnya jaringan tersebut mempelajari fitur yang spesifik terkait dataset pada lapisan-lapisan terakhirnya untuk dapat membedakan kelas-kelas pada data. Sebagai contoh, pada dataset citra, lapisan awal pada jaringan *deep learning* akan belajar untuk mengenali struktur atau fitur umum pada citra seperti komponen bentuk atau garis. Fitur tersebut kemudian dipilah dan diintegrasi hingga dapat digunakan untuk membedakan antara satu kelas dengan kelas lainnya pada lapisan-lapisan akhir jaringan *deep learning*. Hal inilah yang dijadikan sebagai dasar dari metode *transfer learning*.

Transfer learning merupakan metode *machine learning* di mana jaringan belajar menyelesaikan suatu tugas baru dengan mengambil informasi dari model lain yang telah dilatih sebelumnya untuk menyelesaikan tugas lain. Pada *transfer learning*, terdapat dua dataset yang terlibat. Dataset pertama merupakan dataset umum berukuran sangat besar yang biasanya tidak secara langsung berkaitan dengan tugas yang ingin dilakukan. Dataset ini kemudian digunakan untuk proses *training* suatu jaringan *deep learning* (*base model*) untuk mempelajari struktur dasar dan variasi dari jenis data tersebut. Sebagai contoh, salah satu dataset yang biasa digunakan untuk melakukan klasifikasi citra adalah dataset ImageNet yang terdiri dari 1.000 kelas, 1.281.167 citra *training*, 50.000 citra validasi, dan 100.000 citra *testing* [21]. Citra yang terdapat pada dataset ImageNet adalah objek sehari-hari. Proses *training* terhadap dataset pertama ini menghasilkan suatu model yang disebut sebagai *pre-trained* model. Lapisan-lapisan awal pada *pre-trained* model ini berisi informasi mengenai bagaimana mengenali berbagai macam fitur dan struktur pada jenis data tersebut, yang dalam hal ini adalah data citra.

Dataset kedua yang digunakan pada proses *transfer learning* adalah dataset terkait tugas sepesifik yang akan dilakukan. Biasanya dataset kedua ini berukuran kecil sehingga kurang memungkinkan untuk memperoleh performa yang baik apabila digunakan langsung untuk melatih model *deep learning*. Lapisan atas dari *pre-trained* model dibuang dan kemudian diganti dengan lapisan baru (*head model*) yang kemudian akan dilatih pada dataset kedua ini. Hal ini karena lapisan atas *pre-trained* model berisi informasi untuk membedakan kelas-kelas pada dataset pertama, yang tidak dibutuhkan karena tugas yang hendak dilakukan adalah melakukan klasifikasi pada dataset kedua.

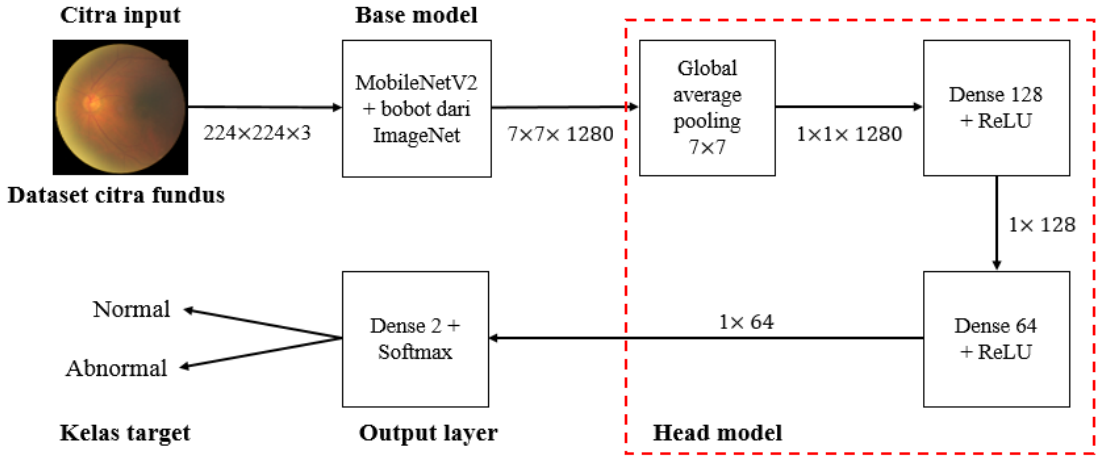
Pada proses *training* menggunakan dataset kedua, lapisan-lapisan bawah pada *pre-trained* model, yang disebut sebagai *base model*, dibekukan *(frozen*). Dengan demikian, lapisan-lapisan pada *head model* hanya bertugas untuk melakukan *feed forward* dan proses pembaruan nilai bobot (*back-propagation*) tidak terjadi pada *base model*. Hal ini karena struktur umum dari suatu jenis data (seperti citra), yang ditangkap oleh *base model*, tetap sama apapun dataset yang digunakan, sehingga proses *training* hanya perlu dilakukan pada *head model* yang bertugas untuk mengenali fitur yang membedakan antar kelas pada dataset kedua. Namun, beberapa lapisan pada *base model* juga dapat dicairkan sehingga dapat dilakukan proses *training* pada lapisan tersebut. Strategi ini disebut sebagai *fine tuning*. *Fine tuning* dilakukan agar jaringan dapat mempelajari struktur yang lebih spesifik dari dataset kedua dan bukan hanya menggunakan struktur umum dari jenis data tersebut.

2.4 MobileNetV2

Pada penelitian ini, digunakan arsitektur MobileNetV2 [20] sebagai *base model*. MobileNetV2 menggunakan *depthwise separable convolution*, selain proses konvolusi standar, untuk meningkatkan efisiensi dari jaringan. Sebuah citra berwarna dua dimensi 2D merupakan matriks tiga dimensi yang terdiri dari dimensi panjang, lebar, dan kedalaman. Pada citra *input*, dimensi ketiga ini biasanya digunakan untuk *channel* warna, di mana terdapat 3 buah *channel* untuk citra RGB (*red green blue*). Sebagai contoh, sebuah citra berwarna dengan ukuran 200 × 200 piksel merupakan sebuah matriks dengan ukuran 200 × 200 × 3 piksel. Proses konvolusi standar memproses semua *input channel* untuk menghasilkan sebuah *output channel* dengan melakukan proses konvolusi pada dimensi ketiga (*depth*) juga. Proses *depthwise separable convolution* terdiri dari dua proses, yaitu *depthwise convolution* yang diikuti dengan *pointwise convolution*. *Depthwise convolution* membagi *input* dan kernel / filter konvolusinya ke dalam *channel depth* yang terpisah dan kemudian melakukan konvolusi pada setiap *channel input* dengan *channel* filter konvolusi yang sesuai. Setelah dihasilkan *output* dari masing-masing *channel*, semua *output channel* tersebut kemudian ditumpuk menjadi satu. *Pointwise convolution* melakukan konvolusi pada *output* proses *depthwise convolution* dengan filter berukuran 1 × 1 piksel untuk menggabungkan tumpukan *output channel* tersebut menjadi satu buah *channel* saja. Proses *depthwise separable convolution* menghasilkan *output* yang sama dengan proses konvolusi standar, namun proses ini melibatkan jumlah parameter yang lebih sedikit sehingga lebih efisien [11].

2.5 Metode Usulan

Arsitektur dari metode yang diusulkan ditunjukkan pada Gambar 3. Pada penelitian ini, *base model* yang digunakan adalah MobileNetV2 yang telah dilatih sebelumnya (*pre-trained*) pada dataset ImageNet [21]. *Base model* ini mengambil *input* berupa citra berukuran 244 × 244 × 3 piksel, sehingga dilakukan penyesuaian ukuran citra pada dataset citra fundus. Arsitektur MobileNetV2 terdiri dari beberapa lapis konvolusi standar dengan ukuran kernel 3 × 3 piksel dan menghasilkan 32 *output channel*. Lapisan ini diikuti oleh beberapa *residual bottleneck layer* yang mengimplementasikan proses *depthwise separable convolution* dengan ukuran kernel 3 × 3 piksel. Di bagian akhir dari *base model* ini terdapat lapisan *pointwise convolution* yang menghasilkan *output* berukuran 7 × 7 × 1280 piksel [20].



Gambar 3. Arsitektur metode yang diusulkan

Di atas lapisan *base model*, pada penelitian ini dikembangkan suatu arsitektur jaringan untuk *head model*. Pada arsitektur asli dari MobileNetV2, lapisan *pointwise convolution* tersebut diikuti oleh lapisan terakhir yang melakukan proses *global average pooling*. Namun pada proses *transfer learning*, lapisan tersebut diganti dengan *head model* usulan yang sesuai dengan dataset yang akan digunakan. *Head model* yang diusulkan terdiri dari lapisan *global*

average pooling dengan ukuran kernel 7 × 7 piksel sehingga menghasilkan *output* berukuran 1 × 1 × 1280 piksel. Lapisan tersebut kemudian diikuti oleh sebuah lapisan *fully-connected* (*dense layer*) yang memetakan *input* 1280 *nodes* ke dalam 128 *nodes* dengan fungsi aktivasi ReLU [14]. Berikutnya terdapat sebuah lagi *dense layer* yang memetakan *input* 128 *nodes* ke dalam 64 *nodes* dengan fungsi aktivasi ReLU. Di bagian akhir terdapat *output layer* yang memprediksi *input* berupa 64 *nodes* ke dalam 2 kelas, yaitu citra fundus "Normal" atau "Abnormal", dengan fungsi aktivasi *softmax*.

Pada penelitian ini diusulkan arsitektur *head model* tersebut karena proses *global pooling* tidak menggunakan banyak parameter yang perlu dilatih sehingga dengan arsitektur ini efisiensi jaringan dapat tetap dipertahankan. Digunakan dua lapis *dense layer* agar jaringan memiliki lebih banyak ruang untuk mempelajari struktur dari dataset citra fundus yang digunakan. Selain itu, arsitektur *head model* yang diusulkan dapat memberikan hasil akurasi yang paling baik berdasarkan hasil uji coba yang dilakukan.

Pada proses *training*, selanjutnya dilakukan perhitungan nilai *error* / *loss* / *cost* dari proses *training* dengan cara membandingkan antara *output* prediksi sistem dengan label data menggunakan *cost function* berupa fungsi *binary cross entropy*. Untuk melakukan proses perubahan bobot jaringan (*back-propagation*), digunakan fungsi optimasi Adam's Optimizer [22] karena fungsi tersebut dapat membantu proses *training* untuk segera mencapai kondisi konvergen [12]. Kondisi konvergen adalah apabila tidak ada perubahan nilai *error* yang berarti pada iterasi proses *training* saat ini dibandingkan dengan iterasi proses *training* sebelum-sebelumnya, sehingga proses *training* bisa diakhiri karena bobot jaringan sudah mencapai nilai optimal. Dengan mencapai kondisi konvergen lebih cepat, maka pada proses *training* dapat digunakan nilai *epoch* (nilai maksimal dilakukannya iterasi proses *training*) berukuran kecil. Pada penelitian ini digunakan nilai *epoch* = 20.

Pada penelitian ini digunakan nilai *input batch* sebanyak 32 citra. Hal ini berarti bahwa ada 32 citra *training* (1 *batch*) yang diproses sekaligus baru kemudian nilai *error* dari ke-32 citra tersebut yang digunakan untuk melakukan proses *back-propagation*. Apabila proses pembaruan bobot jaringan untuk satu *batch* tersebut selesai, maka *batch* (32 citra) berikutnya yang dimasukkan ke dalam jaringan, dst. Satu iterasi dinyatakan selesai apabila seluruh citra *training* telah selesai diproses. Pada penelitian ini terdapat 13 buah *batch* citra *training*. Selain itu, pada penelitian ini digunakan augmentasi data untuk meningkatkan kemampuan generalisasi dari model *deep learning* yang dihasilkan. Generalisasi adalah kemampuan model untuk mengenali data yang tidak dipelajari sama sekali. Augmentasi data melakukan transformasi kepada citra dataset yang digunakan, seperti rotasi, *zoom*, *flip*, *shift*, dsb. Hasil augmentasi dari suatu citra ini kemudian digunakan untuk menggantikan citra tersebut pada setiap iterasi *proses training*. Dengan demikian, pada tiap iterasi tidak digunakan citra-citra yang sama persis, melainkan memiliki sedikit variasi sehingga model yang dihasilkan memiliki kemampuan generalisasi yang lebih baik.

3. HASIL DAN PEMBAHASAN

Penelitian ini diimplementasikan menggunakan bahasa pemrograman Python dan *library* Keras. Eksperimen dilakukan menggunakan perangkat GPU NVIDIA GeForce GTX 860M, RAM 2 × 8GB 2400 MHz DDR4. Performa dari metode-metode yang diuji coba dievaluasi menggunakan nilai akurasi, *precision*, *recall*, dan F1-*score*. Akurasi menunjukkan persentase data yang berhasil diklasifikasi dengan benar oleh sistem terhadap keseluruhan data *testing*. *Precision* menunjukkan persentase data yang berhasil diklasifikasi dengan benar sebagai anggota suatu kelas oleh sistem terhadap jumlah seluruh data yang diklasifikasikan sebagai kelas tersebut oleh sistem. *Recall* menunjukkan persentase data yang berhasil diklasifikasi dengan benar sebagai anggota suatu kelas oleh sistem terhadap jumlah seluruh data *testing* kelas tersebut. F1-*score* mengukur rata-rata *precision* dan *recall* menggunakan Persamaan (3). Semakin tinggi nilai akurasi, *precision*, *recall*, dan F1-*score* dari suatu metode maka semakin baik performa dari metode tersebut. Selain itu, pada penelitian ini juga diukur lama *running time*

dari proses *training* dan *testing* model untuk membandingkan tingkat efisiensi dari model tersebut.

Metode yang diusulkan memberikan nilai akurasi sebesar 72%, *precision* sebesar 72%, *recall* sebesar 72%, F1-*score* sebesar 72%, dan *running time* sebesar 417.59 detik. Secara umum, hasil tersebut cukup baik karena sulit untuk membedakan antara kelas "Normal" dan "Abnormal" pada citra fundus. Hal ini karena variasi tingkat keparahan serta jenis dari penyakit mata yang terdapat pada dataset. Apabila penyakit mata tersebut masih dalam stadium rendah, maka sangat sulit membedakan antara citra kelas "Normal" dengan "Abnormal". Selain itu, antara satu jenis penyakit mata dengan penyakit mata lainnya memiliki ciri-ciri yang berbeda. Citra fundus yang ditampilkan pada Gambar 1 merupakan contoh dari penyakit mata dengan stadium tinggi sehingga cukup tampak perbedaan antara citra kelas "Normal" dengan "Abnormal". Metode yang diusulkan juga dilatih pada dataset yang berukuran relatif kecil, yaitu 601 data. Selain itu, metode yang diusulkan juga memberikan hasil yang lebih baik dibandingkan dengan arsitektur CNN lainnya, seperti yang diuraikan pada eksperimen-eksperimen berikut.

3.1 Eksperimen pada Base Model

Pada eksperimen ini dilakukan perbandingan arsitektur *base model* yang digunakan untuk *transfer learning*. Arsitektur yang dibandingkan adalah MobileNetV2 [20] (metode usulan), ResNet50V2 [17], InceptionV3 [18], InceptionResNetV2 [19], VGG16 [16], dan VGG19 [16]. Arsitektur *head model* yang digunakan pada eksperimen ini sama dengan arsitektur *head model* yang diusulkan. Berdasarkan hasil uji coba yang ditampilkan pada Tabel 1, tampak bahwa performa terbaik diberikan oleh metode usulan yang menggunakan MobileNetV2 sebagai *base model*. Selain itu, arsitektur *MobilNetV2* sebagai *base model* juga memberikan nilai *running time* terkecil yang menunjukkan bahwa arsitektur ini adalah yang paling efisien dibandingkan dengan arsitektur lainnya [11].

Tabel 1 Eksperimen pada *Base Model*

Arsitektur Base Model	**Performa Sistem (%)**				**Running Time (detik)**
	Akurasi	**Precision**	**Recall**	**F1-Score**	
MobileNetV2 [20] **(Usulan)**	**72**	**72**	**72**	**72**	**347.08**
ResNet50V2 [17]	61	65	61	59	1015.01
InceptionV3 [18]	65	65	65	65	639.81
InceptionResNetV2 [19]	65	71	64	62	1440.23
VGG16 [16]	69	70	69	69	2706.28
VGG19 [16]	70	70	70	70	3235.31

3.2 Eksperimen pada Head Model

Pada eksperimen ini dilakukan perbandingan arsitektur *head model* untuk menentukan arsitektur yang memberikan performa terbaik. Pertama-tama, digunakan arsitektur *head model* yang sama dengan arsitektur asli dari MobileNetV2, yaitu terdapat satu lapisan *global average pooling* dengan kernel berukuran 7 × 7 piksel yang langsung terhubung dengan *output layer*. Kemudian dilakukan pengembangan arsitektur *head model* dengan menambahkan dua buah lapisan *fully-connected* yang secara berturut-turut terdiri dari 128 dan 64 *nodes*. Arsitektur ini memberikan performa yang paling baik baik sehingga digunakan pada model yang diusulkan. Selanjutnya dilakukan uji coba menggunakan arsitektur *head model* yang diusulkan oleh Rokhana, et al. (2021) [23] di mana lapisan *global average pooling* diganti dengan lapisan *depthwise convolution* dengan ukuran kernel 3 × 3 piksel. Ternyata performa dari *head model* tersebut tidak sebaik performa *head model* yang diusulkan. Selain itu juga dilakukan uji coba *head model* di mana digunakan lapisan *depthwise separable convolution* dengan kernel 3 × 3 piksel. Performa dari *head model* ini mirip dengan performa dari *head model* yang menggunakan *depthwise convolution*, hanya saja model dengan *depthwise separable convolution* memberikan *running time* yang lebih tinggi karena pada proses tersebut juga terdapat proses *pointwise convolution*. Hasil dari eksperimen ini ditunjukkan pada Tabel 2.

Tampak bahwa arsitektur *head model* yang diusulkan dapat memberikan *running time* yang paling rendah sehingga merupakan arsitektur paling efisien dibandingkan dengan yang lainnya.

Tabel 2 Eksperimen pada *Head Model*

Arsitektur Head Model	**Performa Sistem (%)**				**Running Time (detik)**
	Akurasi	**Precision**	**Recall**	**F1-Score**	
Global average pooling 7x7 [20]	66	66	66	66	352.22
Global average pooling 7x7 – Dense 128 – Dense 64 **(Usulan)**	**72**	**72**	**72**	**72**	**347.08**
Depthwise convolution 3×3 – Dense 128 – Dense 64 [23]	69	69	69	68	360.08
Depthwise separable convolution 3×3×320 – Dense 128 – Dense 64	69	69	68	68	380.22

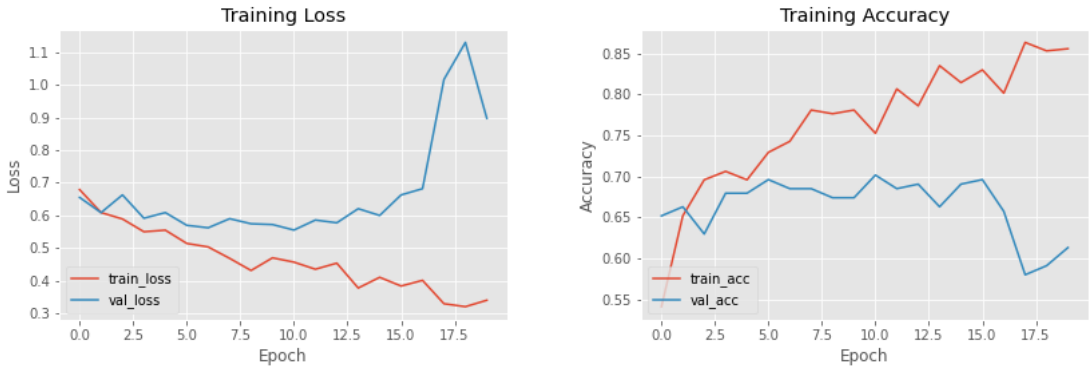
.2 Eksperimen dengan Fine Tuning

Pada penelitian ini juga dilakukan eksperimen dengan melakukan proses *fine tuning* pada *transfer learning*. Pada proses *fine tuning*, terdapat beberapa lapisan pada *base model* yang dicairkan sehingga bobotnya ikut diubah selama proses *training*. Pada eksperimen ini, lapisan yang dicairkan adalah mulai dari *block residual bottleneck* ke-12 hingga lapisan paling akhir. Dilakukan eksperimen *fine tuning* terhadap arsitektur *head model* yang berbeda-beda. Hasil eksperimen dengan *fine tuning* ditunjukkan pada Tabel 3. Apabila dibandingkan dengan Tabel 2, tampak bahwa terjadi penurunan performa model apabila dilakukan *fine tuning*.

Berdasarkan analisis terhadap grafik nilai *error* (*loss*) dan akurasi dari data *testing* selama proses *training* seperti yang ditunjukkan pada Gambar 4, tampak bahwa model yang di-*fine tuning* mengalami *overfitting*. *Overfitting* terjadi apabila model terlalu sesuai dengan data *training* sehingga memiliki kemampuan generalisasi yang rendah dan gagal mengklasifikasi data baru. *Overfitting* dapat diketahui dengan menganalisis grafik *error* maupun akurasi dari proses *training* model, di mana model yang *overfitting* akan memiliki nilai *loss* yang turun atau akurasi yang naik untuk data *training* (garis merah pada Gambar 4), namun memiliki nilai *loss* yang naik atau nilai akurasi yang turun untuk data *testing* (garis biru pada Gambar 4) seiring dengan bertambahnya iterasi proses *training*.

Tabel 3 Eksperimen dengan *Fine Tuning*

Arsitektur Head Model	**Performa Sistem (%)**				**Running Time (detik)**
	Akurasi	**Precision**	**Recall**	**F1-Score**	
Global average pooling 7x7	61	70	61	56	556.71
Global average pooling 7x7 – Dense 128 – Dense 64	**64**	67	**64**	**63**	**552.96**
Depthwise convolution 3×3 – Dense 128 – Dense 64	61	68	61	57	570.13
Depthwise separable convolution 3×3×320 – Dense 128 – Dense 64	63	**74**	63	58	588.72



(a) (b)

Gambar 4. Grafik nilai (a) *loss* dan (b) akurasi dari model dengan arsitektur *head model* ke-3 (depthwise convolution 3×3 – dense 128 – dense 64) yang di-*fine tuning*

Analisis terhadap nilai *recall* dari masing-masing kelas ("Normal" dan "Abnormal") menunjukkan bahwa model yang di-*fine tuning* hanya dapat mengenali salah satu jenis kelas saja dan gagal mengenali jenis kelas yang lain. Sebagai contoh, arsitektur usulan (Global average pooling 7x7 – Dense 128 – Dense 64) yang di-*fine tuning* memberikan nilai *recall* 83% untuk kelas "Normal" dan 45% untuk kelas "Abnormal". Hal ini menunjukkan bahwa model yang di-*fine tuning* ini cenderung mengelompokkan data yang masuk ke dalam kelas "Normal". Oleh sebab itu, pada penelitian ini tidak dilakukan proses *fine tuning* pada metode usulan.

4. KESIMPULAN DAN SARAN

Pada penelitian ini diusulkan metode berbasis *Convolutional Neural Network* (CNN) untuk melakukan deteksi penyakit mata pada citra fundus. Metode yang diusulkan menggunakan *transfer learning* dan arsitekturnya terdiri dari 2 bagian, yaitu *base model* dan *head model*. Berdasarkan hasil eksperimen, arsitektur MobileNetV2 digunakan sebagai *base model* karena efisien dan memberikan akurasi yang lebih tinggi dibandingkan dengan arsitektur CNN lainnya, seperti ResNet50V2, InceptionV3, InceptionResNetV2, VGG16, dan VGG19. Arsitektur *head model* yang diusulkan, yang terdiri dari lapisan *global average pooling* dan diikuti oleh 2 lapisan *fully-connected*, mampu memberikan akurasi yang paling tinggi dan efisiensi paling baik dibandingkan dengan arsitektur *head model* lainnya. Eksperimen pada dataset citra fundus yang terdiri dari 601 citra dan terbagi dalam dua kelas ("Normal" dan "Abnormal") menunjukkan bahwa metode yang diusulkan mampu memberikan performa yang baik pada dataset berukuran relatif kecil yang berisi berbagai jenis penyakit mata dengan nilai akurasi sebesar 72%, *precision* sebesar 72%, *recall* sebesar 72%, F1-*score* sebesar 72%, dan *running time* sebesar 417.59 detik. Hasil uji coba menunjukkan bahwa tidak perlu dilakukan proses *fine tuning* pada model karena dapat menyebabkan *overfitting* dan menurunkan akurasi dari model. Untuk penelitian lebih lanjut, dapat dilakukan pengembangan untuk meningkatkan performa model dalam mendeteksi penyakit mata, seperti dengan memodifikasi arsitektur *head model* yang telah diusulkan, menambahkan *layer dropout*, dsb. Selain itu, juga dapat dikembangkan model untuk mendeteksi jenis spesifik dari berbagai macam penyakit mata.

DAFTAR PUSTAKA

[1] G. P. Kumasela, J. S. M. Saerang, and L. Rares, "Hubungan Waktu Penggunaan Laptop Dengan Keluhan Penglihatan Pada Mahasiswa Fakultas Kedokteran Universitas Sam Ratulangi," *J. e-Biomedik*, vol. 1, no. 1, 2013, doi: 10.35790/ebm.1.1.2013.4361.

[2] The International Agency for the Prevention of Blindness (IAPB), "Vision Atlas," *The International Agency for the Prevention of Blindness (IAPB)*, 2020. https://www.iapb.org/learn/vision-atlas/magnitude-and-projections/.

[3] M. J. Burton *et al.*, "The Lancet Global Health Commission on Global Eye Health: vision beyond 2020," *Lancet Glob. Heal.*, vol. 9, no. 4, pp. e489–e551, 2021, doi: 10.1016/S2214-109X(20)30488-5.

[4] N. Gour, M. Tanveer, and P. Khanna, "Challenges for ocular disease identification in the era of artificial intelligence," *Neural Comput. Appl.*, vol. 5, 2022, doi: 10.1007/s00521-021-06770-5.

[5] V. Gulshan *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *JAMA - J. Am. Med. Assoc.*, vol. 316, no. 22, pp. 2402–2410, 2016, doi: 10.1001/jama.2016.17216.

[6] A. Diaz-Pinto, S. Morales, V. Naranjo, T. Köhler, J. M. Mossi, and A. Navea, "CNNs for automatic glaucoma assessment using fundus images: An extensive validation," *Biomed. Eng. Online*, vol. 18, no. 1, pp. 1–19, 2019, doi: 10.1186/s12938-019-0649-y.

[7] L. Li, M. Xu, X. Wang, L. Jiang, and H. Liu, "Attention based glaucoma detection: A large-scale database and CNN model," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 10563–10572, 2019, doi:

10.1109/CVPR.2019.01082.

[8] M. N. Bajwa *et al.*, "Correction to: Two-stage framework for optic disc localization and glaucoma classification in retinal fundus images using deep learning (BMC Medical Informatics and Decision Making (2019) 19 (136) DOI: 10.1186/s12911-019-0842-8)," *BMC Med. Inform. Decis. Mak.*, vol. 19, no. 1, pp. 1–16, 2019, doi: 10.1186/s12911-019-0876-y.

[9] A. K. Gangwar and V. Ravi, *Diabetic Retinopathy Detection Using Transfer Learning and Deep Learning*, vol. 1176. Springer Singapore, 2021.

[10] R. C. Joshi, M. K. Dutta, P. Sikora, and M. Kiac, "Efficient Convolutional Neural Network Based Optic Disc Analysis Using Digital Fundus Images," *2020 43rd Int. Conf. Telecommun. Signal Process. TSP 2020*, pp. 533–536, 2020, doi: 10.1109/TSP49548.2020.9163560.

[11] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv Prepr. arXiv1704.04861*, 2017, [Online]. Available: http://arxiv.org/abs/1704.04861.

[12] R. Indraswari, T. Kurita, A. Z. Arifin, N. Suciati, and E. R. Astuti, "Multi-projection deep learning network for segmentation of 3D medical images," *Pattern Recognit. Lett.*, vol. 125, pp. 791–797, Jul. 2019, doi: 10.1016/j.patrec.2019.08.003.

[13] R. Indraswari, R. Rokhana, and W. Herulambang, "Melanoma image classification based on MobileNetV2 network," *Procedia Comput. Sci.*, vol. 197, pp. 198–207, 2021, doi: 10.1016/j.procs.2021.12.132.

[14] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," *arXiv Prepr. arXiv1803.08375*, no. 1, pp. 2–8, 2018, [Online]. Available: http://arxiv.org/abs/1803.08375.

[15] R. Rokhana, W. Herulambang, and R. Indraswari, "Deep Convolutional Neural Network for Melanoma Image Classification," *IES 2020 - International Electronics Symposium: The Role of Autonomous and Intelligent Systems for Human Life and Comfort*. pp. 481–486, 2020, doi: 10.1109/IES50839.2020.9231676.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9908 LNCS, pp. 630–645, 2016, doi: 10.1007/978-3-319-46493-0_38.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 2818–2826, 2016, doi: 10.1109/CVPR.2016.308.

[19] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," *31st AAAI Conf. Artif. Intell. AAAI 2017*, pp. 4278–4284, 2017.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4510–4520, 2018, doi: 10.1109/CVPR.2018.00474.

[21] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 2009, pp. 248–255.

[22] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[23] R. Rokhana, W. Herulambang, and R. Indraswari, "Multi-Class Image Classification Based on MobileNetV2 for Detecting the Proper Use of Face Mask," *Int. Electron. Symp. 2021 Wirel. Technol. Intell. Syst. Better Hum. Lives, IES 2021 - Proc.*, pp. 636–641, 2021, doi: 10.1109/IES53407.2021.9594022.