Pengaruh *Preprocessing* Terhadap Klasifikasi *Diabetic Retinopathy* dengan Pendekatan *Transfer Learning Convolutional Neural Network*

http://dx.doi.org/10.28932/jutisi.v7i1.3327
Riwayat Artikel
Received: 24 Januari 2021 | *Final Revision*: 25 Februari 2021 | *Accepted*: 4 Maret 2021

Juan Elisha Widyaya[#1], Setia Budi✉[*2]

[#]*Magister Ilmu Komputer, Universitas Kristen Maranatha*
Jl. Prof. Drg. Surya Sumantri, MPH 65. Bandung
[1]mi1979006@student.it.maranatha.edu
[2]setia.budi@it.maranatha.edu

Abstract —**Diabetic retinopathy (DR) is eye diseases caused by diabetic mellitus or sugar diseases. If DR is detected in early stages, the blindness that follows can be prevented. Ophthalmologists or eye clinicians usually decide the stage of DR from retinal fundus images. Careful examination of retinal fundus images is a time-consuming task and requires experienced clinicians or ophthalmologists but a computer which has been trained to recognize the DR stages can diagnose and give results in real-time manner. One approach of algorithms to train a computer to recognize an image is deep learning Convolutional Neural Network (CNN). CNN allows a computer to learn the features of an image, in our case is retinal fundus image, automatically. Preprocessing is usually done before a CNN model is trained. In this study, four preprocessing were carried out on "APTOS 2019 Blindness Detection" dataset. Of the four preprocessing tested, preprocessing with Contrast Limited Adaptive Histogram Equalization followed by unsharp masking on the green channel of the retinal fundus image give the best results with an accuracy of 78.79%, 82.97% precision, 74.64% recall, and 95.81% AUC. The CNN architecture used is Inception v3.**

Keywords— **classification; Convolutional Neural Network; Diabetic Retinopathy; Inception v3; transfer learning**

I. PENDAHULUAN

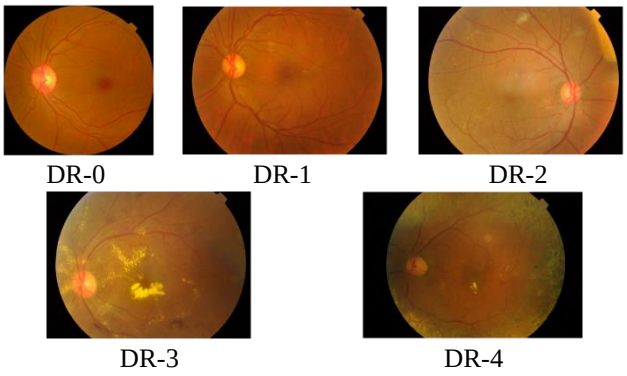
Diabetic Retinopathy (DR) atau yang dikenal juga sebagai *Diabetic Eye Disease* adalah penyakit mata akibat komplikasi diabetes yang merusak retina dan dapat mengakibatkan kebutaan [1]. DR merupakan penyakit progresif yang akan berkembang semakin parah seiring lamanya seorang pasien menderita DR. Deteksi DR sejak dini sangat penting agar pasien mendapatkan perawatan untuk mengurangi perkembangan DR dan mencegah kebutaan karena pada tahap tertentu DR tidak bisa disembuhkan lagi [2]. Diagnosis pasien DR biasanya dilakukan oleh *ophthalmologist* dengan memeriksa citra *fundus* retina secara seksama. Proses pemeriksaan seperti ini cukup memakan waktu [3]. Belum lagi perbandingan antara jumlah pasien potensial dan ketersediaan tenaga medis tidak seimbang, sehingga mendorong kebutuhan adanya sistem untuk membantu deteksi DR secara otomatis [2].

Deteksi DR pada citra *fundus* retina dapat dibagi menjadi dua tugas [2], 1) deteksi DR berdasarkan tingkat *lesion* dan 2) deteksi DR berdasarkan tingkat citra. Deteksi pada tingkat *lesion* artinya mendeteksi gejala klinis pada citra *fundus* retina, seperti tanda-tanda *microaneurysm*, *hemorrhages*, *exudates*, dan *neovascular*. Deteksi DR pada tingkat citra artinya deteksi DR pada citra *fundus* retina untuk menilai apakah terdapat tanda-tanda DR.

Deteksi pada tingkat citra mengklasifikasikan citra *fundus* ke dalam dua kategori, yaitu normal atau memiliki tanda-tanda DR. Citra *fundus* yang terdapat tanda-tanda DR dapat dibagi ke dalam dua kelas berdasarkan tingkat keparahannya 1) *Non-Proliferative Diabetic Retinopathy* (NPDR) dan 2) *Proliferative Diabetic Retinopathy* (PDR). NPDR adalah tahap awal di mana diabetes mulai merusak pembuluh darah kecil pada retina, hal ini umum ditemui pada penderita diabetes; di mana pembuluh darah mulai mengeluarkan cairan dan darah yang menyebabkan retina bengkak. Seiring berjalannya waktu, pembengkakan atau *edema* dapat menebalkan retina yang menyebabkan pandangan kabur. Fitur klinis pada tahap ini setidaknya terdapat satu *microaneurysm* atau *hemorrhage* baik tanpa atau dengan *exudates* [2].

PDR adalah tahapan lanjut yang mengarah ke pertumbuhan pembuluh darah baru. PDR ditandai dengan penyebaran pembuluh darah abnormal di dalam retina menuju rongga *vitreous*. Pembuluh darah baru ini bersifat rapuh dan dapat mengakibatkan pendarahan pada rongga *vitreous* yang menyebabkan kebutaan. Terdapat beberapa

sistem untuk menilai tingkat keparahan DR, seperti *American Academy of Ophthalmology* (AAO), *Early Treatment of Diabetic Retinopathy Study* (ETDRS), dan protokol penilaian Scottish. Pada penelitian ini DR diklasifikasikan menjadi lima kelas yang mengacu pada protokol penilaian Scottish, yaitu tidak ada DR, DR ringan, DR sedang, DR parah, dan PDR. Contoh citra *fundus* retina dengan tingkat DR yang berbeda dapat dilihat pada Gambar 1. Gambar kiri-atas dengan label DR-0 adalah citra *fundus* retina tidak ada DR, atau citra *fundus* retina normal. Gambar kanan-bawah dengan label DR-4 adalah citra *fundus* retina PDR.



Gambar 1. Citra *fundus* retina dengan tingkat DR yang berbeda

Berbagai upaya dilakukan untuk mendeteksi dan mengklasifikasikan citra *fundus* retina dengan bantuan komputer atau *Computer Aided Diagnosis* (CAD) [2]. Salah satu pendekatan untuk mengklasifikasikan DR pada citra *fundus* retina adalah dengan pendekatan *deep learning Convolutional Neural Network* (CNN). CNN berkembang pesat dan banyak digunakan untuk tugas klasifikasi citra sejak kompetisi ImageNet pada tahun 2012 yang dimenangkan oleh Alex Krizhevsky dan kawan-kawan [4]. Mereka membawa revolusi CNN melalui penggunaan *Graphics Processing Unit* (GPU), *Rectified Linear Units* (ReLU), regularisasi *dropout*, dan augmentasi data yang efektif [5].

Training model CNN dapat dilakukan dengan salah satu dari dua pendekatan, yaitu *end to end learning* atau *transfer learning*. *Training* model dengan pendekatan *end to end learning* memiliki tantangan sendiri [5], yaitu membutuhkan jumlah data yang besar. Namun jumlah data yang besar tidak selalu dijumpai pada domain citra medis. Jika terdapat dataset dalam jumlah yang besar, permasalahan berikutnya muncul, yaitu kebutuhan akan sumber daya komputasi yang besar. Selain itu, *training* model CNN juga sering kali sulit karena *overfitting* atau masalah konvergensi model sehingga dibutuhkan penyesuaian *hyperparameter* model berulang kali.

Salah satu pendekatan alternatif adalah *training* model CNN dengan pendekatan *transfer learning*. Pendekatan *transfer learning* menggunakan model yang sudah di-*training* pada domain dataset tertentu (umumnya dataset ImageNet [6]) lalu disesuaikan (*fine-tuned*) pada domain dataset baru (dataset citra *fundus* retina). Menurut Yosinski dan kawan-kawan [7], dan Nima Tajbakhsh dan kawan-kawan [5], pendekatan *transfer learning* cenderung memberikan hasil yang lebih baik jika dibandingkan dengan pendekatan *end to end learning*.

Sebelum model CNN di-*training*, biasanya akan dilakukan *preprocessing* terhadap citra *fundus* retina untuk mengatasi beberapa tantangan pada pengolahan citra digital seperti variasi skala, variasi iluminasi, atau gambar tidak fokus. Dari tinjauan pustaka yang dilakukan, tidak banyak yang membahas pengaruh *preprocessing* terhadap performa model. Penelitian ini membandingkan pengaruh *preprocessing* terhadap performa model diantaranya adalah Maria A. Bravo [8] dan Chunyan Lian [9]. Dari tinjauan pustaka juga dapat dilihat bahwa pendekatan *transfer learning* lebih banyak digunakan jika dibandingkan dengan *end to end learning*. Hal ini umum dilakukan karena keterbatasan jumlah citra untuk *training* model CNN menjadi salah satu tantangan di bidang citra medis [5]. Namun *fine tuning* pada pendekatan *transfer learning* tersebut tidak membahas berapa banyak lapisan dari arsitektur CNN yang perlu di-*fine tuning*. Salah satu penelitian yang menyampaikan hal ini adalah penelitian yang dilakukan oleh Saboora dan kawan-kawan [10]. Saboora melakukan *fine tuning* sebanyak dua dan empat blok pada arsitektur Inception v3 dan Xception untuk klasifikasi biner citra *fundus* retina.

Sesuai dengan latar belakang yang disampaikan, maka penelitian ini menguji pengaruh *preprocessing* terhadap performa model dan menguji banyaknya blok Inception v3 yang perlu di-*fine tuning*. Tugas klasifikasi yang diberikan adalah mengklasifikasikan citra *fundus* retina ke dalam lima kelas. Sehingga terdapat dua pertanyaan yang dijawab pada penelitian ini, yaitu:

- Apa pengaruh *preprocessing* terhadap hasil *training* model dengan pendekatan *transfer learning*?
- Berapa banyak blok Inception v3 dari sebuah model yang di-*training* dengan pendekatan *transfer learning* perlu di-*fine tuning*?

Diharapkan dari penelitian ini bisa memberikan jawaban atas dua rumusan masalah diatas, yaitu:

- Mengetahui pengaruh *preprocessing* yang diterapkan pada citra *fundus* retina terhadap performa model yang dihasilkan.
- Mengetahui jumlah blok Inception v3 yang perlu di-*fine tuning* ketika *training* model dengan pendekatan *transfer learning* untuk tugas klasifikasi citra *fundus* retina.

Penelitian ini membangun model untuk mengklasifikasikan citra *fundus* retina menjadi lima kelas, mulai dari kelas tidak ada DR, DR ringan, DR sedang, DR parah, dan PDR dengan teknik CNN. Arsitektur CNN yang dipakai adalah Inception v3 [11]. Dataset yang digunakan adalah citra *fundus* retina yang diambil dari Kaggle.com [12]. Dataset ini merupakan dataset untuk kompetisi pada Kaggle

dengan judul "APTOS 2019 Blindness Detection". Penelitian ini fokus untuk melihat pengaruh *preprocessing* terhadap model yang dihasilkan dan melihat jumlah blok Inception v3 yang perlu di-*fine tuning* dengan pendekatan *transfer learning*.

Penelitian ini mengemukakan dua hipotesis sebagai berikut:

- *Preprocessing* yang tepat akan memberikan dampak positif terhadap performa model. Dengan *preprocessing* diharapkan fitur yang diinginkan pada citra menjadi lebih menonjol.
- *Fine tuning* perlu dilakukan pada beberapa blok terakhir saja dari model yang di-*training* dengan pendekatan *transfer learning*. Pada lapisan awal, *fine tuning* tidak memberikan pengaruh terhadap performa model karena pada lapisan awal CNN mengekstrak fitur umum dari citra.

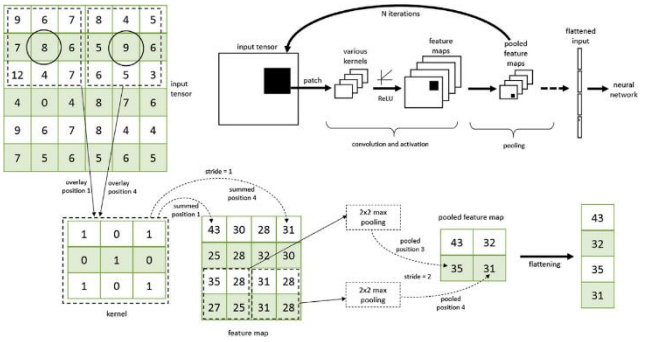
II. LANDASAN TEORI

Convolutional Neural Network (CNN) adalah *Neural Network* (NN) khusus untuk memproses data yang memiliki topologi berbentuk *grid* [13]. Data *time series* adalah data dengan topologi satu dimensi dan citra adalah data dengan topologi dua dimensi. Nama *Convolutional Neural Network* mengindikasikan bahwa *Neural Network* melakukan proses operasi matematika *convolution* yang merupakan operasi *linear*. *Convolutional Neural Network* secara sederhana adalah *Neural Network* yang menggunakan *convolution* sebagai operasi multiplikasi matrix setidaknya pada salah satu lapisan. CNN terdiri dari lapisan *convolution* dan *pooling* yang memungkinkan fitur diekstrak dari citra.

A. *Convolutional Neural Network*

CNN mengekstrak fitur dari citra dengan melakukan operasi *convolution* antara *kernel* dengan *input tensor* yang merupakan *subset array pixel* [14]. Perubahan posisi *kernel* secara berturut-turut dikenal dengan istilah *strides*. *Stride* lebih dari satu memberikan efek *downsampling* pada *output* operasi *convolution*. Produk perkalian *element wise* dari setiap *subset array pixel* dan *kernel* akan dijumlahkan untuk menghasilkan satu nilai numerik dalam *feature map*. *Feature map* ini diteruskan ke fungsi aktivasi, yang biasanya menggunakan fungsi aktivasi *Rectified Linear Unit* (ReLU) sebelum diteruskan ke lapisan *pooling*. Lapisan *pooling* melakukan *downsample* terhadap *feature map* untuk mengurangi kebutuhan daya komputasi dan mencegah *overfitting*. Operasi *pooling* yang digunakan umumnya adalah *max pooling*. *Max pooling* menghasilkan *output* sama dengan nilai maksimum dari setiap *patch* data pada *feature map*. *Max pooling* dengan *kernel* 2×2 dan *stride* 2 akan mengambil satu nilai maksimum dari setiap empat *feature map*. Sekuensial dari *convolution*, *kernel*, dan *pooling* menghasilkan sejumlah lapisan data atau fitur yang di bagian akhir lapisan akan ditransformasi menjadi *array* satu dimensi. Proses transformasi ini disebut dengan istilah *flattening*.

Gambar 2 adalah contoh operasi CNN. CNN memiliki sejumlah lapisan *convolutional* dan *pooling* yang menghasilkan *feature map* sebelum tahap *flattening* dan masuk ke *Neural Network* untuk klasifikasi. Pada gambar tersebut *input* adalah *tensor* dua dimensi 6×6 dan *convolutional* menggunakan *kernel* 3×3 yang berjalan secara sekuensial dengan *stride* 1. Operasi *convolution* menghasilkan *feature map* berukuran 4×4. Lapisan *max pooling* menggunakan kernel berukuran 2×2 dengan *stride* 2 yang mengambil satu nilai maksimum dari empat elemen terkait (*patch*) sebagai representasi data. Operasi *max pooling* menghasilkan *pooled feature map* berukuran 2×2. *Pooled feature map* kemudian diubah dari dua dimensi menjadi satu dimensi (*flattened*) sebelum masuk ke *Neural Network* [14].

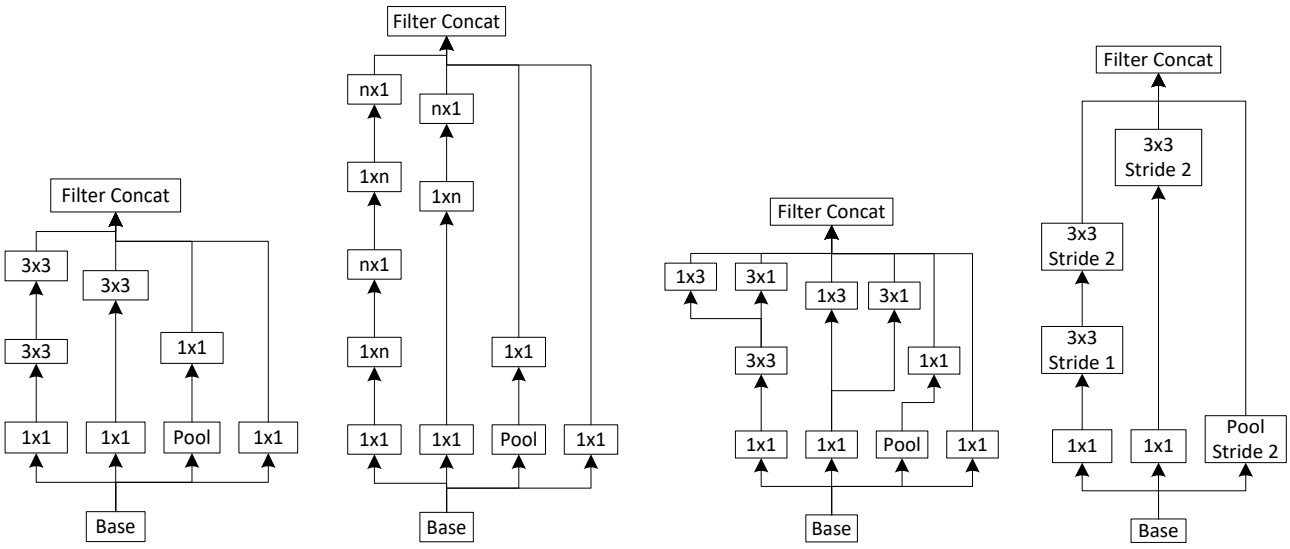


Gambar 2. Proses *Convolutional Neural Network* [14]

B. *Inception v3*

Arsitektur Inception v1 dari GoogLeNet didesain untuk bekerja dengan baik dengan kondisi komputasi yang terbatas [15]. Hal ini dicapai dengan melakukan penggabungan *output* dari *kernel convolution* berukuran 1×1, 3×3, 5×5 dan *pooling* sambil menjaga tinggi dan lebar *output* dari setiap *kernel* tidak berubah [15], [16]. Lapisan *bottleneck* atau operasi *convolution* 1×1 selalu dilakukan sebelum operasi *convolution* 3×3 dan 5×5 untuk mengurangi jumlah operasi matematika sampai dengan faktor 10. Arsitektur ini dikembangkan lebih lanjut oleh Christian Szegedy dan kawan-kawan [11] menjadi Inception v3.

Arsitektur Inception v3 [11] memanfaatkan teknik faktorisasi *convolution* asimetris untuk mengurangi jumlah operasi matematika. Modul Inception v3 terdiri atas tiga desain yaitu [17]:



Modul A | Modul B | Modul C | *Grid Size Reduction*

Gambar 3. Modul arsitektur Inception v3 [11]

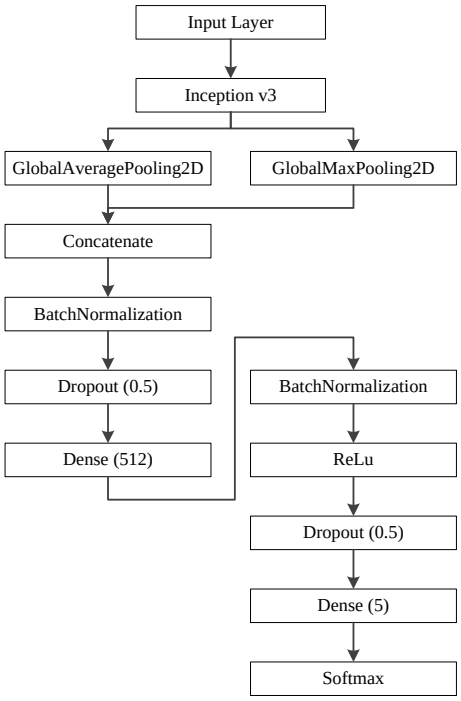
- Modul Inception A (Gambar 3 Modul A): modul ini mengganti *convolution* 5×5 pada Inception v1 dengan dua buah *convolution* 3×3. Jika ditinjau dari total operasi matematika, *convolution* 5×5 membutuhkan 25 operasi matematika, sedangkan dua buah *convolution* 3×3 membutuhkan 18 operasi matematika.
- Modul Inception B (Gambar 3 Modul B): modul ini menggunakan teknik *convolution* asimetris 1×7 dan 7×1 sebagai ganti *convolution* 7×7.
- Modul Inception C (Gambar 3 Modul C): modul ini diajukan untuk mempromosikan representasi dimensi tinggi.

Inception v3 menawarkan langkah paralel (Gambar 3 *Grid Size Reduction*) yang menggabungkan operasi *convolution* dan operasi *pooling* untuk mendapatkan jumlah *feature map* yang sama dengan operasi *convolution* diikuti *pooling* atau pun sebaliknya namun *Grid Size Reduction* memiliki beban komputasi yang lebih ringan.

Secara umum arsitektur Inception v3 dapat dibagi menjadi dua bagian besar, yaitu bagian *convolution* dan bagian *classifier* yang digambarkan pada Gambar 15. Bagian *convolution* adalah kumpulan dari lapisan *convolution*, *pooling*, aktivasi ReLU, dan *Batch Normalization* yang akan mengekstrak fitur dari citra *input*. Bagian *classifier* tersusun atas *Neural Network* yang bertugas mengklasifikasikan fitur yang sudah diekstrak oleh bagian *convolution.*

C. *Transfer Learning*

Ketika *training* model dengan pendekatan *transfer learning*, bagian *classifier* dari *pretrained* model perlu diganti dengan *classifier* baru. *Classifier* yang ditambahkan pada bagian akhir CNN di-*training* terlebih dahulu sementara bobot pada bagian *convolution* akan dibekukan. Setelah *classifier* konvergen, *fine tuning* dapat dilakukan pada bagian *convolution*. Menurut Yosinski dan kawan-kawan [7], *fine tuning* dari lapisan manapun pada bagian *convolution* memberikan *benefit* terhadap performa model.



Gambar 4. Arsitektur *classifier* yang dimodifikasi

Arsitektur yang ditambahkan pada bagian akhir CNN dapat dilihat pada Gambar 4. Arsitektur ini terinspirasi dari Fastai namun dimodifikasi pada posisi BatchNormalization dan Relu. Pada Fastai [18], BatchNormalization diletakkan setelah lapisan aktivasi Relu. Namun menurut Andrew Ng

[19], BatchNormalization umumnya diletakkan sebelum lapisan aktivasi ReLU.

D. *Penelitian Terkait*

Berbagai penelitian sudah dilakukan untuk membangun model menggunakan komputer untuk mendiagnosis DR secara otomatis [2]. Metode tradisional *machine learning* akan mengekstrak fitur dari citra *fundus* retina. Fitur yang sudah diekstrak tersebut menjadi *input* bagi *classifier* tertentu seperti *random forest*, *support vector machine*, atau *AdaBoost classifier* [20]. Salah satu prasyarat teknik *hand engineered feature* adalah memahami gejala klinis DR secara mendalam untuk memilih fitur yang tepat. Berbeda dengan *deep learning* seperti CNN yang akan mengekstrak fitur secara otomatis dari citra [21]. Jika dibandingkan dengan *hand engineered feature*, teknik *deep learning* secara umum memberikan hasil yang lebih unggul [2].

Maria A. Bravo dan Pablo A. Arbelaez [8] menggunakan dataset dari EyePACS untuk mengklasifikasikan DR ke dalam lima kelas. *Preprocessing* yang dilakukan adalah:

- Circle RGB: citra *fundus* retina RGB yang dipotong melingkar dengan ukuran 224×224 *pixel*.
- Square: citra *fundus* retina yang dipotong bujur sangkar dengan ukuran 224×224 *pixel*.
- Color centered: citra circle RGB dikurangi dengan warna rata-rata lokal menggunakan filter Gaussian. Intensitas setiap *channel* citra dipusatkan pada 127.
- Grayscale: citra circle RGB dikonversi menjadi citra *grayscale*.

TABEL I
HASIL PERCOBAAN MARIA A. BRAVO

Preprocessing	**VGG-16**	**Inception V4**
Circle RGB	46.3%	44.62%
Square	45.3%	
Color centered	48.3%	45.00%
Grayscale	48.1%	

Arsitektur CNN yang digunakan adalah VGG-16 dan Inception v4. Kedua arsitektur ini di-*training* dengan pendekatan *transfer learning*. Model VGG-16 di-*training* kira-kira sebanyak 14 *epoch*, *learning rate* 0.001 yang diubah setiap 5 *epoch* dengan faktor 10, *batch size* 115, optimasi *stochastic gradient descent with momentum* (SGDM) (momentum = 0.9), dan regularisasi *dropout* untuk mengurangi *overfitting*. Sedangkan model Inception v4 di-*training* sebanyak 20 *epoch*, *batch size* 32, *learning rate* 0.001, dan regularisasi *weight decay* 0.00004. Selama *training* model, baik arsitektur VGG-16 maupun arsitektur Inception v4, dilakukan augmentasi citra dengan rotasi antara 0° sampai dengan 360°, perbesar dan perkecil gambar antara 0 sampai dengan 20 *pixel*, pembalik vertikal dan horizontal yang dilakukan secara acak. Hasil percobaan ini bisa dilihat pada Tabel I.

Saboora Mohammadian dan kawan-kawan [10] menggunakan dataset EyePACS untuk mengklasfikasikan DR ke dalam dua kelas. *Preprocessing* yang dilakukan adalah mengubah ukuran citra agar memiliki ukuran yang sama, nilai setiap *pixel* dikurangi dengan nilai rata-rata lokal, lalu citra dipetakan terhadap 50% abu-abu sehingga ketajaman citra lebih merata. Pinggiran dari citra retina dipotong untuk menghilangkan efek garis batas. Arsitektur yang dipilih adalah Inception v3 dan Xception yang di-*training* dengan pendekatan *transfer learning*. Model di-*training* sebanyak 200 iterasi dengan nilai *learning rate* 0.0001. Augmentasi yang digunakan adalah pergeseran citra, rotasi citra, pembalikan citra vertikal dan horizontal selama *training*. Saboora membandingkan beberapa parameter *training* untuk melihat pengaruhnya terhadap performa model yang dihasilkan. Parameter yang dibandingkan adalah optimasi *Adaptive Moment Estimation* (ADAM) dengan SGDM (momentum = 0.9), aktivasi *Exponential Linear Unit* (ELU) dengan ReLU, dan dua blok dengan empat blok yang di-*fine tuning*. Hasil terbaik dengan nilai akurasi 87.12% diperoleh dari model Inception v3, optimasi ADAM, dan *fine tuning* sebanyak dua blok.

Gabriel Garcia dan kawan-kawan [22] menggunakan dataset EyePACS untuk mengklasifikasikan DR ke dalam dua kelas. *Preprocessing* yang dilakukan adalah mengubah ukuran citra sehingga memiliki ukuran yang sama, citra dikurangi dengan nilai rata-rata lokal, lalu dipetakan ke abu-abu. Setelah *preprocessing* tersebut, citra diskalakan menjadi 256×256 *pixel*. Augmentasi yang dilakukan adalah *flipping* citra dan *cropping* 80% citra. Gabriel Garcia menggunakan arsitektur VGG-16 yang di-*training* dengan pendekatan *transfer learning*. Karena jumlah citra dari masing-masing tidak seimbang, Gabriel menggunakan *class weight* dengan perbandingan untuk kelas normal dan DR sebesar 1:2.74. Model VGG di-*training* dengan regularisasi *weight decay* 0.00005. Performa model yang diperoleh adalah akurasi 83.68%, *sensitivity* 54.47%, *specificity* 93.65%.

Xiaoliang Wang dan kawan-kawan [16] menggunakan dataset dari EyePACS yang dipilih 166 citra saja untuk mengklasifikasikan DR ke dalam lima kelas. *Preprocessing* yang dilakukan mengubah ukuran citra menjadi 227×227 untuk AlexNet, 224×224 untuk VGG-16, dan 299×299 untuk Inception v3. Ketiga arsitektur ini di-*training* dengan pendekatan *transfer learning*. Parameter *training* yang digunakan adalah: optimasi SGDM (momentum = 0.9), fungsi aktivasi ReLU, dan beberapa parameter lain yang berbeda untuk setiap arsitektur. Hasil akurasi yang diperoleh adalah 37.43% untuk AlexNet, 50.03% untuk VGG-16, dan 63.23% untuk Inception v3.

Chunyan Lian dan kawan-kawan [9] menggunakan dataset EyePACS untuk mengklasifikasikan DR ke dalam lima kelas. *Preprocessing* yang mereka lakukan adalah mengubah citra menjadi 256×256 *pixel*, meningkatkan warna citra dengan mengurangi warna rata-rata lokal, dan membuang warna latar hitam. Pada dataset dilakukan *subsampling* dan augmentasi untuk mengatasi persoalan dataset yang tidak seimbang. Terdapat tiga arsitektur yang dicoba, yaitu AlexNet, ResNet-

50, dan VGG-16. Ketiga arsitektur ini di-*training* dengan pendekatan *transfer learning*. Parameter yang digunakan selama *training* adalah: *learning rate* 0.001 yang berkurang dengan faktor 10 setiap 27 *epoch*, *batch size* 25, optimasi SGDM (momentum = 0.9), dan *weight decay* 0.0005. Hasil akurasi yang diperoleh pada penelitian ini adalah 73.19% untuk AlexNet, 76.41% untuk ResNet-50, 79.04% untuk VGG-16 dengan *preprocessing*, dan 76.01% untuk model VGG-16 tanpa *preprocessing*.

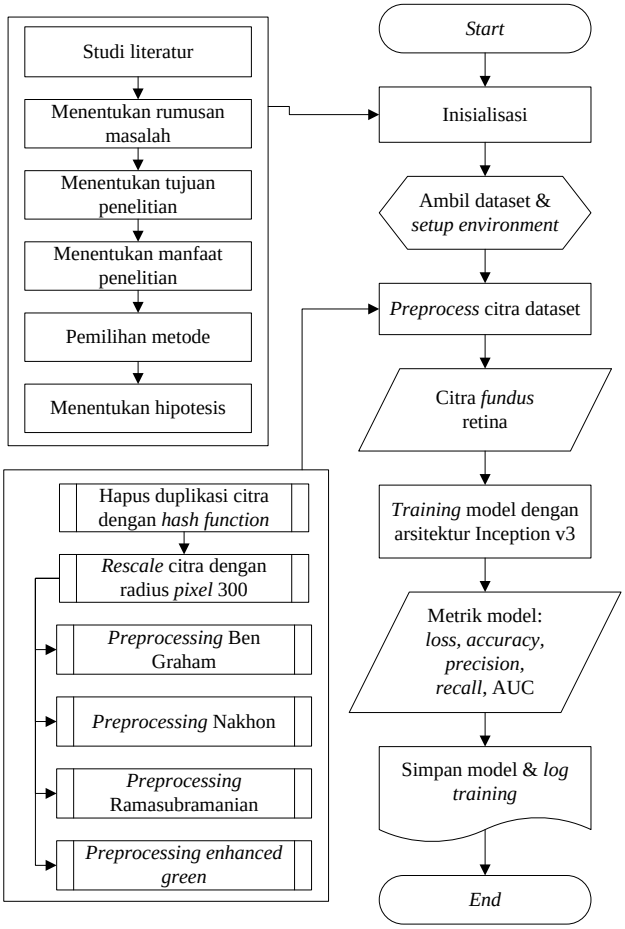
Zhentao Gao dan kawan-kawan [20] menggunakan dataset yang dibangun sendiri dengan total 4,476 citra untuk mengklasifikasikan DR menurut perawatan yang disarankan. DR diklasifikasikan ke dalam empat kelas, dari normal, sedang, berat, dan parah. *Preprocessing* yang dilakukan adalah mentransformasi citra, sehingga setiap citra memiliki ukuran dan bentuk yang sama, dan melakukan normalisasi warna. Augmentasi citra selama *training* adalah pembalikan citra horizontal dan vertikal, rotasi citra antara [-25º, 25º] secara acak, pembesaran citra antara [0.85, 1.15], dan distorsi citra secara acak. Terdapat lima arsitektur yang dicoba pada penelitian Zhentao Gao, yaitu ResNet-18, Resnet-101, VGG-19, Inception v3, dan arsitektur Inception@4 yang merupakan modifikasi dari Inception v3. Beberapa parameter yang digunakan untuk *training* model adalah: *batch size* 32, optimasi ADAM, *learning rate* 0.00001, *weight decay* 0.2, dan aktivasi ReLU. Akurasi yang diperoleh adalah 87.61% untuk ResNet-18, 87.26% untuk ResNet-101, 85.50% untuk VGG-19, 88.35% untuk Inception v3, dan 88.72% untuk Inception@4.

Misgina Tsighe Hagos dan kawan-kawan [23] menggunakan dataset EyePACS untuk mengklasifikasikan DR ke dalam dua kelas. Jumlah dataset yang digunakan terdiri dari 2,500 citra *training* dan 5,000 citra *validation*. *Preprocessing* yang mereka lakukan adalah membuang warna latar hitam, mengubah citra menjadi 300×300 *pixel*, dan mengurangi warna rata-rata lokal dari setiap *pixel*. Arsitektur CNN yang digunakan adalah Inception v3 yang di-*training* dengan pendekatan *transfer learning*. Parameter yang dipakai selama *training* adalah: optimasi SGD dengan *ascending learning rate* 0.0005, fungsi *loss cosine*, dan fungsi aktivasi ReLU. Hasil akurasi yang diperoleh adalah 90.9%.

Md Sazzad Hossen dan kawan-kawan [1] menggunakan dataset APTOS-2019 untuk mengklasifikasikan DR ke dalam lima kelas. *Preprocessing* yang mereka lakukan adalah mengubah skala citra sehingga memiliki radius yang sama, mengurangi warna rata-rata lokal yang dipetakan ke 50% *grayscale*, memotong citra menjadi ukuran 90%. Ukuran citra yang digunakan untuk *training* adalah 224×224 *pixel*. Arsitektur CNN yang dipilih adalah DenseNet-121 dan di-*training* dengan pendekatan *transfer learning*. Selama *training* mereka melakukan augmentasi dengan *flipping* horizontal dan vertikal secara acak dan pembesaran antara 85% sampai dengan 115% secara acak. Parameter *training* yang digunakan adalah: optimasi ADAM, total *epoch* 12, regularisasi *dropout*, dan fungsi aktivasi ReLU. Hasil yang didapatkan dari dataset *validation* adalah akurasi 0.9491, precision 0.9598, *recall* 0.9256, F-*score* 0.9395, dan AUC 0.9852.

III. METODOLOGI PENELITIAN

Secara umum penelitian ini mengklasifikasikan citra *fundus* retina ke dalam lima kelas berdasarkan tingkatannya. Citra *fundus* retina tersebut diklasifikasikan ke dalam salah satu kelas: DR-0, DR-1, DR-2, DR-3, dan DR-4. Penelitian ini mengimplementasi teknik *deep learning* CNN yang mengekstrak fitur secara otomatis dari sebuah citra. Arsitektur CNN yang digunakan adalah Inception v3 dengan modifikasi pada bagian *classifier* seperti pada Gambar 4.



Gambar 5. Diagram alir penelitian

A. *Pipeline Penelitian*

Pipeline penelitian ini dapat dilihat pada Gambar 5. Penelitian diawali dengan studi literatur, menentukan rumusan masalah, tujuan penelitian, pemilihan metode, dan hipotesis yang diuji. Eksperimen diawali dengan membuang duplikasi citra dalam dataset [24], lalu semua citra diubah ukurannya sehingga memiliki radius *pixel* 300 pada wilayah retina [25]. Citra yang sudah diubah ukurannya ini

dilanjutkan dengan empat *preprocessing* terpisah, yaitu *preprocessing* Ben Graham [25], Nakhon [26], Ramasubramanian [27], dan *enhanced green*.

Setelah *preprocessing*, dilakukan *training* model dengan menggunakan teknik *deep learning* CNN. *Training* model dilakukan dengan pendekatan *transfer learning* menggunakan beberapa kondisi dataset, yaitu:

- Citra tanpa *preprocessing* hanya dilakukan *rescale*.
- Citra dengan *preprocessing* Ben Graham [25].
- Citra dengan *preprocessing* Nakhon Ratchasima [26].
- Citra dengan *preprocessing* Ramasubramanian [27]
- Citra dengan *preprocessing enhanced green*.
- Citra tanpa *preprocessing* hanya dilakukan *rescale* lalu di-*fine tuning* sebanyak *n* blok Inception v3, n = [1, 2, …, 10] dan semua lapisan.

Pada bagian akhir *training*, model dievaluasi berdasarkan metrik *loss*, akurasi, *precision*, *recall*, dan AUC. Meskipun terdapat beberapa metrik yang digunakan, metrik akurasi dipilih sebagai metrik utama untuk mengukur performa model.

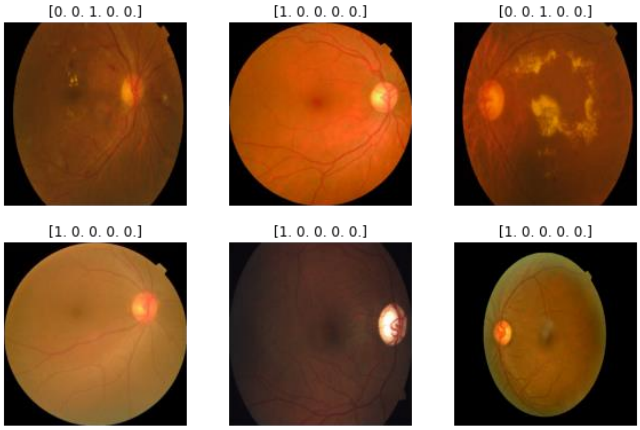
B. *Dataset*

Dataset yang digunakan untuk *training* model diambil dari dari Kaggle.com [12]. Dataset ini berasal dari rumah sakit mata Aravind Eye Hospital [13] yang dikeluarkan bersamaan dengan kegiatan APTOS Symposium ke-4 [28]. Dataset APTOS 2019 ini diberikan kepada publik dengan tujuan untuk meningkatkan kemampuan rumah sakit mengidentifikasi tingkatan DR dengan bantuan teknologi dan dapat memberikan hasil yang konsisten atau *robust* dengan berbagai kondisi dan variasi citra *input*. Dataset merupakan fotografi *fundus* retina dalam berbagai kondisi yang telah diberi label terhadap setiap citra dengan skala 0 sampai 4 oleh ahli klinis seperti yang dapat dilihat pada Tabel II.

TABEL II
LABEL UNTUK CITRA SESUAI DENGAN TINGKAT DR

Label	Label *one hot*	Indikasi	Jumlah citra
0	[1, 0, 0, 0, 0]	Tidak ada DR	1,805
1	[0, 1, 0, 0, 0]	DR rendah	370
2	[0, 0, 1, 0, 0]	DR sedang	999
3	[0, 0, 0, 1, 0]	DR parah	193
4	[0, 0, 0, 0, 1]	DR *proliferative*	270
Total citra			**3,662**

Kumpulan citra dalam dataset ini merupakan citra sebagaimana adanya ketika diambil yaitu terdapat *noise* pada citra dan label. Di antara citra terdapat objek asing, tidak fokus, kurang cahaya atau kelebihan cahaya. Citra ini juga diambil dari berbagai klinik menggunakan beragam kamera di waktu yang berbeda sehingga memberikan variasi yang tinggi terhadap kumpulan citra. Contoh dataset dapat dilihat pada Gambar 6. Pada gambar tersebut dapat terlihat citra *fundus* retina dengan labelnya yang *encoded* dalam bentuk *one hot vector*.

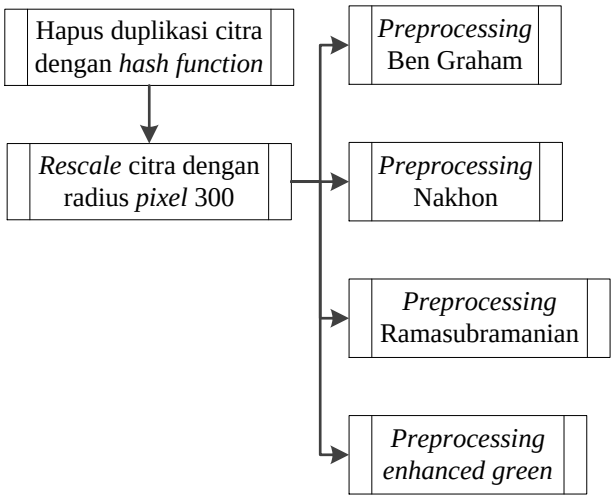


Gambar 6. Contoh citra *fundus* retina beserta dengan labelnya.

Deskripsi dataset adalah sebagai berikut:

- *File format* citra: PNG
- Variasi dimensi citra: $474 \times 358\ pixel$ – $3{,}388 \times 2{,}588\ pixel$
- Variasi ukuran *file* citra: 217 kB – 7,495 kB
- Jumlah citra *training*: 3,662 citra

C. *Preprocessing Dataset*



Gambar 7. Diagram alir *preprocessing*

Preprocessing diawali dengan memeriksa apakah terdapat duplikasi citra dalam dataset. Citra yang memiliki duplikasi akan dieliminasi dari dataset. Dataset yang sudah bersih dari duplikasi akan diubah ukurannya sehingga memiliki radius *pixel* yang sama untuk area retina pada citra. Berdasarkan tinjauan pustaka, *preprocessing* yang umum digunakan pada citra *fundus* retina merujuk pada *preprocessing* Ben Graham. Selain *preprocessing* tersebut terdapat *preprocessing* yang ditawarkan oleh Nakhon Ratchasima [26], yang merupakan *notebook preprocessing* dengan *vote* terbanyak pada Kaggle kompetisi dengan judul "APTOS 2019 Blindness Detection". Ramasubramanian [27] juga mengemukakan teknik *preprocessing* dengan menggunakan *channel* hijau saja. *Preprocessing* Ramasubramanian menjadi dasar inspirasi

untuk *preprocessing enhanced green*. Sehingga terdapat empat *preprocessing* yang diterapkan kepada dataset dan diuji pengaruhnya terhadap performa model, yaitu *preprocessing* Ben Graham [25], Nakhon [26], Ramasubramanian [27], dan *enhanced green*. Hubungan *preprocessing* yang satu dengan yang lain dapat dilihat pada Gambar 7.

1) *Menghapus Duplikasi Citra*

Sebelum masuk ke tahap *preprocessing*, dataset diperiksa terlebih dahulu, apakah terdapat duplikasi citra. Algoritma *difference hash* (*dhash*) digunakan untuk memberikan *signature* atau identitas pada setiap gambar [24]. Dari *signature* tersebut dapat ditemukan duplikasi jika terdapat dua citra dengan *signature* yang sama. *Signature* diperoleh dengan mengambil *n* digit bilangan biner dari sebuah citra. Langkah untuk membuat *signature* dengan algoritma *dhash* adalah:

- Konversi citra RGB menjadi citra *grayscale.*
- Mengubah ukuran citra menjadi ukuran $(n + 1) \times n$ $(column \times row)$.
- Periksa apakah nilai *pixel* ke $i > i + 1$ dalam satu baris citra sehingga menghasilkan $n - 1$ perbedaan antara *pixel* yang bersebelahan. n baris dengan n nilai perbedaan menghasilkan n^2 deret bilangan biner.
- Deret biner `True / False` (1 atau 0) tersebut dikonversikan menjadi *integer* berbentuk desimal yang dirumuskan dengan:

$$signature = \sum_{i=0}^{n-1} d_i \times 2^i \qquad (1)$$

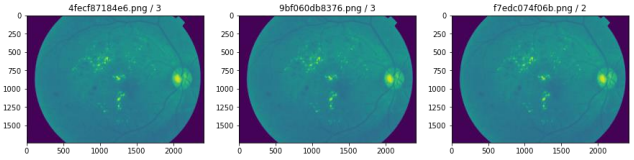
Di mana:

- d_i adalah nilai 1 atau 0 dari deret biner ke - i

Pada penelitian ini n yang dipakai adalah 32, sehingga diperoleh 1,024-bit nilai *hash* yang dikonversi ke bilangan *integer*. Nilai *integer* tersebut yang menjadi *signature* dari sebuah citra. Citra duplikasi dieliminasi dari dataset dengan ketentuan sebagai berikut:

- Jika terdapat dua citra atau lebih dengan *signature* yang sama dan label yang sama, maka diambil satu citra saja.
- Jika terdapat dua citra atau lebih dengan *signature* yang sama dan label yang berbeda, maka citra dikeluarkan dari dataset.

Pada Gambar 8 dapat dilihat duplikasi citra sebanyak tiga buah dengan label yang berbeda. Citra dengan kondisi seperti ini dieliminasi semuanya dari dataset. Label terletak di bagian atas citra.



Gambar 8. Contoh gambar duplikasi dengan label yang berbeda.

Setelah citra dibersihkan dari duplikasi, maka diperoleh jumlah citra yang digunakan adalah 3,498 seperti yang dapat dilihat pada Tabel III.

TABEL III
JUMLAH CITRA SEBELUM DAN SETELAH MENGHAPUS DUPLIKASI

Tingkat DR	Sebelum	Setelah	Komposisi Setelah
0	1,805	1,796	51.3%
1	370	338	9.7%
2	999	921	26.3%
3	193	173	4.9%
4	270	270	7.7%
Total	3,662	3,498	

Dari Tabel III dapat dilihat bahwa kelas 0 memiliki komposisi citra sebanyak 51.3% dari total dataset. Jumlah citra yang tidak seimbang seperti ini akan menghasilkan model yang didominasi oleh salah satu kelas dengan jumlah citra terbanyak. Harry Pratt [29] dan Gabriel Garcia [22] menggunakan pembobotan setiap kelas sesuai dengan jumlah citra dari masing-masing kelas untuk mengatasi jumlah kelas yang tidak seimbang. Pembobotan kelas juga digunakan pada penelitian ini. Bobot masing-masing kelas dihitung dengan cara [30]:

$$w_j = \frac{\sum_i^k n_i}{c \times n_j} \qquad (2)$$

Di mana:

- $\sum_i^k n_i$ adalah jumlah seluruh citra dari k kelas
- w_j adalah bobot kelas ke - j
- n_j adalah jumlah citra dari kelas ke - j
- c adalah jumlah kelas dari dataset

Dengan perhitungan tersebut diperoleh bobot dari setiap kelas yang dapat dilihat pada Tabel IV.

TABEL IV
BOBOT DARI SETIAP KELAS DR

Tingkat DR	Jumlah citra	Bobot kelas
0	1,796	0.389532
1	338	2.069822
2	921	0.759609
3	173	4.043931
4	270	2.591111
Total	**3,498**	

2) *Mengubah Ukuran Citra*

Teknik mengubah ukuran citra mengadopsi teknik yang digunakan Ben Graham [25]. Citra diubah ukurannya sehingga memiliki radius *pixel* yang sama. Radius *pixel* yang digunakan adalah 300 *pixel*. Berikut ini adalah potongan kode dengan bahasa *python:*

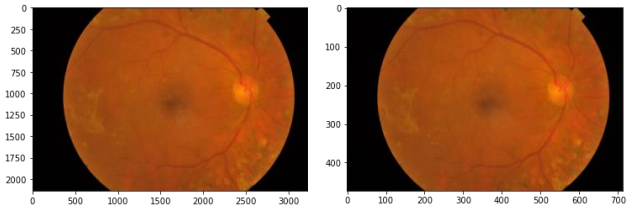
Algoritma 1: Mengubah ukuran citra dengan radius yang ditentukan
Input: `img` dan `scale=300`
Output: `s` (faktor skala citra)

```
x = img[img.shape[0] // 2, :, :].sum(1)
r = (x > x.mean() / 10).sum() / 2
s = scale * 1.0 / r
```

Di mana:

- `x` menghitung nilai citra pada baris tengah dan dijumlahkan pada sumbu *channel*.
- `r` menghitung radius *fundus* retina. Operasi `x > x.mean() / 10` memberikan nilai `True` untuk area retina dan `False` untuk area warna hitam.
- `s` mengkonversi nilai `r` dengan radius *pixel* pilihan yaitu 300 *pixel*. Nilai skalar `s` digunakan sebagai faktor skala untuk mengubah ukuran citra.

Teknik mengubah ukuran citra dengan menggunakan faktor skala akan tetap memperhatikan aspek rasio dari setiap citra. Hasil dari citra yang sudah diubah ukurannya dapat dilihat pada Gambar 9. Citra yang diubah ukurannya ini menjadi dasar untuk setiap *preprocessing* berikutnya karena citra yang ukurannya lebih kecil mengurangi beban komputasi.



Gambar 9. Citra sebelum dan setelah ukuran diubah

3) *Preprocessing Ben Graham*

Terdapat tiga tahapan *preprocessing* yang dilakukan oleh Ben Graham [25], yaitu:

- Mengubah ukuran citra dengan metode *rescale* yang sudah dibahas pada bagian 2) Mengubah Ukuran Citra.
- Mengurangi warna citra dengan warna rata-rata lokal, lalu dipetakan ke 50% abu-abu
- Memotong citra secara melingkar menjadi 90% dari radius *pixel* yang diberikan untuk menghilangkan "*boundary effects*."

Citra dikurangi warna rata-rata lokal dengan menggunakan persamaan klasik linear *unsharp masking* yang diberikan dengan:

$$y(n,m) = \lambda \times x(n,m) + (-\lambda) \times g(n,m) + 128 \qquad (3)$$

Di mana:

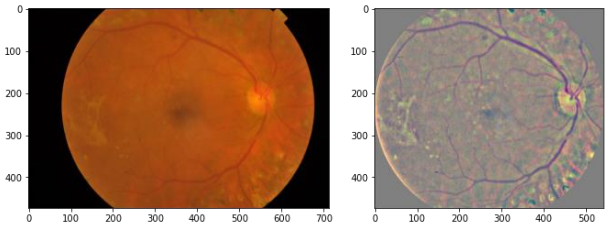
- $y(n,m)$ adalah citra *output*
- $x(n,m)$ adalah citra *input*,
- $g(n,m)$ adalah citra yang diburamkan dengan *Gaussian blur*.
- λ ($\lambda = 4$) adalah faktor skala yang mengontrol peningkatan kontras citra *output*

Nilai $g(n,m)$ diperoleh dari:

$$g(n,m) = G(n,m,\sigma) * x(n,m) \qquad (4)$$

Di mana:

- $G(n,m,\sigma)$ adalah *gaussian filter* dengan $\sigma = 10$
- $*$ adalah operator *convolution*
- $x(n,m)$ adalah citra *input*

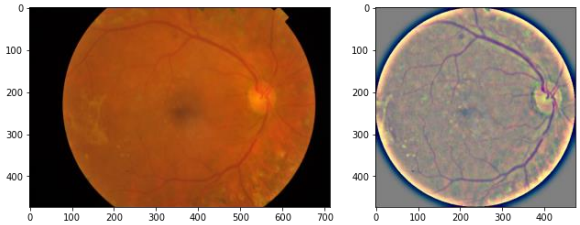


Gambar 10. *Preprocessing* Ben Graham

Hasil dari *preprocessing* Ben Graham dapat dilihat pada Gambar 10. Gambar sebelah kiri merupakan gambar *input*, yang belum melewati *preprocessing* Ben Graham dan gambar sebelah kanan adalah citra yang sudah melewati *preprocessing* Ben Graham. Tampak bahwa warna citra setelah *preprocessing* lebih homogen.

4) *Preprocessing Nakhon*

Preprocessing Nakhon [26] diawali dengan mengubah ukuran citra dan memotong margin hitam pada citra. Margin hitam dipotong dengan mencari baris dan kolom pada citra yang memiliki setidaknya satu pixel di sepanjang baris dan kolom yang lebih besar dari nilai *threshold* yang ditentukan ($threshold = 7$) sehingga membentuk *bounding box* pada area *fundus* retina saja [31]. Setelah margin hitam dibuang, proses selanjutnya serupa dengan *preprocessing* yang dilakukan oleh Ben Graham, namun Nakhon tidak membuang "*boundary effect*". Nakhon memotong setiap citra secara melingkar, sehingga citra *fundus* retina memiliki bentuk yang sama. Gambar 11 menunjukan citra *input* dan citra *output* dari *preprocessing* Nakhon. Bisa dilihat pada citra *output*, bentuk retina dipotong dengan melingkar sempurna dan memiliki "*boundary effect"*

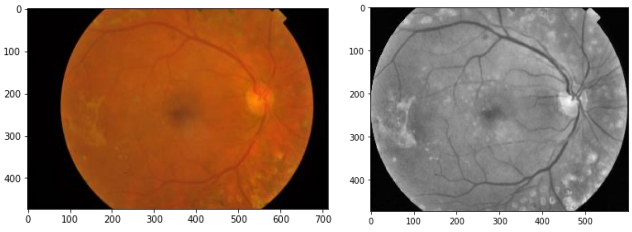


Gambar 11. *Preprocessing* Nakhon Ratchasima

5) *Preprocessing Ramasubramanian*

Ramasubramanian dan Selvaperumal [27] mengusulkan teknik *preprocessing* dengan mengambil *green channel* dari citra RGB karena *green channel* memiliki informasi paling banyak dan menunjukkan diskriminasi antara gejala klinis dengan warna latar belakang. Mula-mula citra diambil *channel* hijau saja lalu margin hitam dipotong. *Median filter* digunakan untuk menghilangkan *salt and pepper noise* pada citra sambil mempertahankan garis tepi pada citra *fundus* retina. Ukuran *kernel* median *filter* yang digunakan adalah 3×3.

Setelah *noise* dihilangkan, citra ditingkatkan dengan menggunakan *Contrast Limited Adaptive Histogram Equalization* (CLAHE). CLAHE membagi citra *input* ke dalam delapan wilayah kontekstual lalu *histogram equalization* diterapkan pada wilayah tersebut. Dengan CLAHE, fitur tersembunyi seperti *exudates*, *microaneurysms*, *fovea*, *blood vessel* akan lebih terlihat. Hasil *preprocessing* usulan Ramasubramanian dapat dilihat pada Gambar 12. Citra hasil *preprocessing* Ramasubramanian disimpan sebagai citra tiga *channel* yang terdiri atas *channel* hijau semua (*Green*, *Green*, *Green* / G, G, G). Dari kiri ke kanan: citra sebelum *preprocessing*, citra yang sudah melewati *preprocessing* Ramasubramanian dan dilipatkan menjadi tiga *channel*.



Gambar 12. *Preprocessing* Ramasubramanian dan Selvaperumal

6) *Preprocessing Enhanced Green*

Preprocessing enhanced green menerapkan dua teknik *processing* citra, yaitu CLAHE yang diikuti dengan *unsharp masking*. Citra yang digunakan pada *preprocessing* ini adalah citra dari *channel* hijau saja seperti yang diusulkan oleh Ramasubramanian. Mula-mula kontras citra ditingkatkan dengan CLAHE. CLAHE membagi citra *input* ke dalam beberapa wilayah kontekstual dengan kernel berukuran 8×8 lalu *histogram equalization* diterapkan pada wilayah tersebut. Menurut Ramasubramanian [27], CLAHE menonjolkan fitur tersembunyi seperti *exudates*, *microaneurysms*, *fovea*, *blood vessel*.

Setelah kontras citra diperbaiki, citra ditajamkan dengan *unsharp masking*. *Unsharp masking* telah lama digunakan dalam industri percetakan dan penerbitan untuk menajamkan citra dengan mengurangi citra asli dengan citra yang sudah diburamkan (*unsharp*). Proses *unsharp masking* terdiri dari beberapa langkah [32]:

- Citra asli diburamkan. Teknik untuk memburamkan citra menggunakan *Gaussian blur* seperti yang diusulkan oleh Ben Graham.
- Kurangi citra asli dengan citra yang sudah diburamkan. Proses ini membentuk *mask*.
- Tambahkan citra asli dengan *mask* yang terbentuk

Unsharp masking bisa diberikan dengan persamaan [33]:

$$y(n,m) = \alpha \times x(n,m) + \beta \times g(n,m) \quad (5)$$

Di mana:

- $y(n,m)$ adalah citra *output*
- $x(n,m)$ adalah citra *input*
- $g(n,m)$ adalah citra yang diburamkan dengan *Gaussian blur*.
- α adalah koefisien citra asli. α ditetapkan dengan nilai 4 mengikuti koefisien yang dipilih oleh Ben Graham.
- β adalah $1 - \alpha$

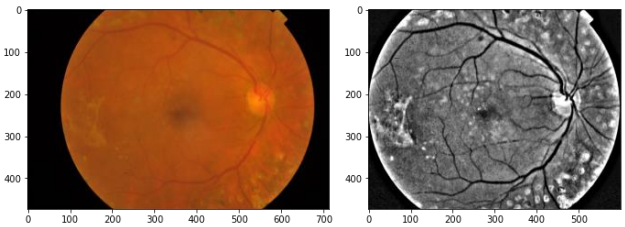
Nilai $g(n,m)$ diperoleh dari:

$$g(n,m) = G(n,m,\sigma) * x(n,m) \quad (6)$$

Di mana:

- $G(n,m,\sigma)$ adalah *gaussian filter* dengan $\sigma = 10$
- $x(n,m)$ adalah citra *input*
- $*$ adalah operator *convolution*

Citra hasil *preprocessing enhanced green* disimpan sebagai citra tiga *channel* yang terdiri atas *channel* hijau semua (G, G, G). Hasil *preprocessing enhanced green* dapat dilihat pada Gambar 13. Citra sebelah kiri merupakan citra *input*. Citra sebelah kanan merupakan citra *output* dari *preprocessing enhanced green* yang dilipatkan menjadi tiga *channel*. Dapat dilihat pada gambar tersebut, objek pada gambar berupa titik atau garis lebih tajam.



Gambar 13. *Preprocessing enhanced green*

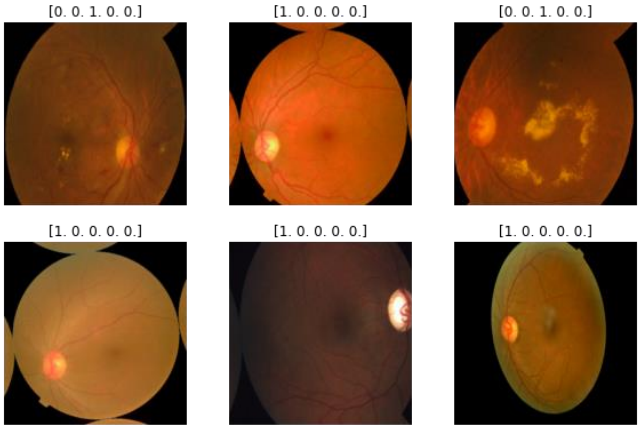
D. *Augmentasi*

Augmentasi citra adalah strategi yang memungkinkan praktisi untuk meningkatkan keragaman data yang tersedia untuk *training* model, tanpa benar-benar mengumpulkan data baru [34]. Dengan augmentasi, model menerima *input* citra yang sudah ditransformasi sehingga citra *input* lebih beragam. Tujuan augmentasi adalah agar model tidak melihat citra yang sama dua kali selama *training*. Augmentasi membantu model terpapar aspek yang lebih banyak dari data

dan menghasilkan model yang lebih mengeneralisir [35]. Teknik augmentasi yang dipakai adalah:

- Rotasi citra: 10°
- Pergeseran citra pada sumbu x dan sumbu y
- Distorsi citra dengan *shear* sebesar 0.1
- *Zoom*: 90% – 110%
- *Flipping* horizontal dan vertikal

Augmentasi ini diterapkan untuk citra *training* saja, sedangkan untuk citra *validation* tidak dilakukan augmentasi. Citra yang diaugmentasi dapat dilihat pada Gambar 14.



Gambar 14. Citra *training* dengan augmentasi

E. *Training Model*

Training model menggunakan *library* Keras, yang merupakan *high level* API dari *platform deep learning* Tensorflow. Pada Keras sudah terdapat berbagai model *pretrained* yang di-*training* dengan dataset ImageNet, termasuk model dengan arsitektur Inception v3. Perangkat keras yang digunakan adalah CPU Core2Quad Q9550, RAM 8GB, GPU GTX 1070 8GB. Adapun *Hyperparameter* yang dipakai untuk *training* model dapat dilihat pada Tabel V.

TABEL V
HYPERPARAMETER YANG DIPAKAI SETIAP *TRAINING* MODEL

No	Hyperparameter	Nilai
1	Ukuran citra	$299 \times 299 \times 3$
2	*Batch Size*	32
3	Optimisasi	ADAM
4	*Loss function*	*Categorical cross entropy*
5	Metrik	*Accuracy* (*Acc*), *precision*, *recall*, AUC
6	*Learning rate classifier*	$1e^{-4}$
7	*Epoch classifier*	50
8	*Learning rate fine tuning*	$2e^{-6}$
9	*Epoch fine tuning*	50
10	*Weight decay*	0.01

Training model terbagi menjadi dua skema sesuai dengan rumusan masalah yang diberikan, yaitu:

- *Training* model untuk menguji pengaruh *preprocessing* terhadap performa model.
- *Training* model untuk menguji banyaknya blok Inception v3 yang perlu di-*fine tuning*.

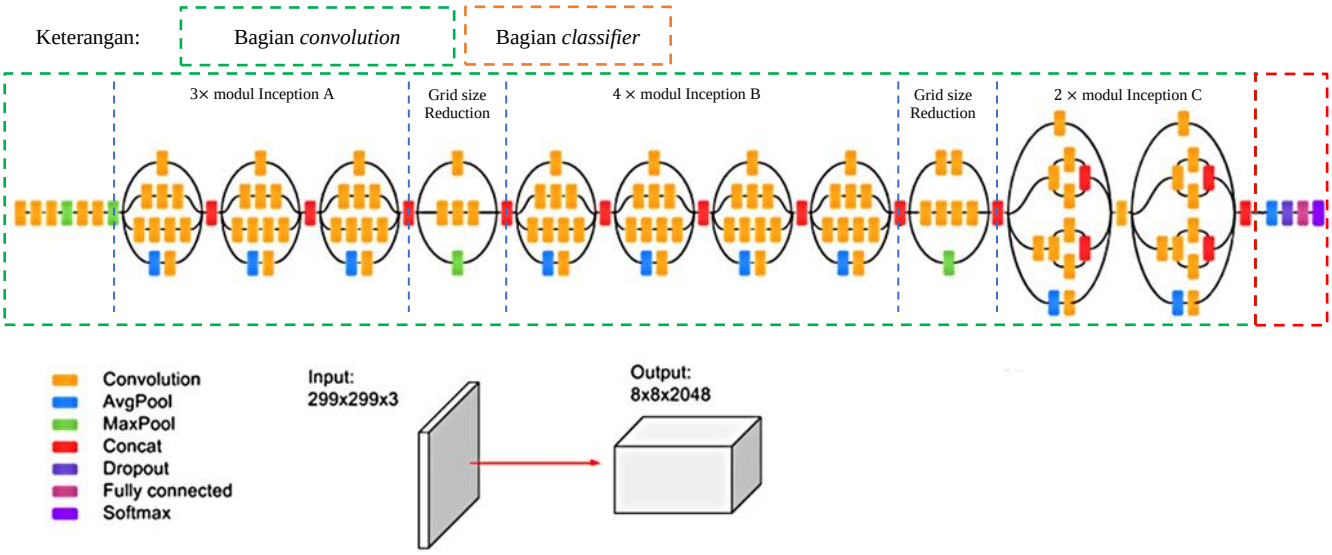
1) *Pengujian Pengaruh Preprocessing*

Training model ini dilakukan dengan pendekatan *transfer learning*. Bagian *convolution* dari Inception v3 menggunakan bobot *pretrained* dari dataset ImageNet. Sedangkan bobot bagian *classifier* diinisialisasi dengan Glorot Uniform [36]. *Training* model dilakukan pada bagian *classifier* terlebih dahulu lalu diikuti dengan *fine tuning* dua blok Inception v3 [10]. *Training classifier* dilakukan sebanyak 50 *epoch* dengan *learning rate* $1e^{-4}$. Sewaktu *training classifier*, bobot pada bagian *convolution* dibekukan sehingga tidak diperbaharui. *Fine tuning* dilakukan sebanyak 50 *epoch* dengan *learning rate* $2e^{-6}$.

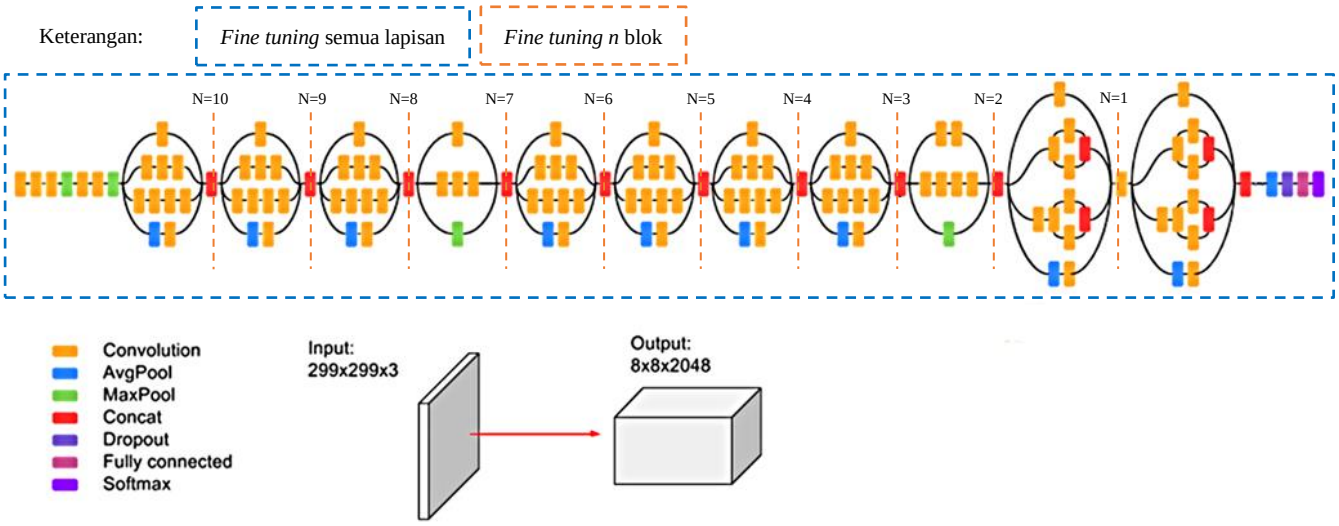
Dataset dibagi menjadi 80% untuk *training* dan 20% untuk *validation* dengan metode *5-fold cross validation*. Hasil dari *5-fold cross validation* ini dirata-ratakan sebagai nilai akhir akurasi model. Dataset yang dipakai pada *training* ini adalah 1) citra yang sudah diubah ukurannya, 2) citra dengan *preprocessing* Ben Graham, 3) citra dengan *preprocessing* Nakhon Ratchasima, 4) citra dengan *preprocessing* Ramasubramanian, dan 5) citra dengan *preprocessing enhanced green.*

2) *Fine Tuning n Blok Inception v3*

Training model ini menggunakan dataset yang diubah ukurannya. Mula-mula *training* dilakukan pada bagian *classifier*, yang diikuti dengan *fine tuning* terhadap blok Inception v3 seperti yang diilustrasikan pada Gambar 16. *Fine tuning* dilakukan sebanyak n blok, $n = [1,2,3, \dots ,10]$ dan semua lapisan. Dataset dibagi menjadi dua, yaitu 80% train dan 20% *validation* dengan teknik *holdout*, bukan dengan 5-*fold cross validation*.



Gambar 15. Bagian umum arsitektur CNN [37]



Gambar 16. *Fine tuning n* blok Inception v3 [37]

IV. HASIL DAN PEMBAHASAN

Hasil *training* untuk melihat pengaruh *preprocessing* terhadap performa model akan fokus pada akurasi model. *Training* dengan lima kondisi dataset menghasilkan lima model, yaitu:

- Model *rescale_png* adalah model yang di-*training* dengan dataset asli yang diubah ukurannya.
- Model *ben_graham* adalah model yang di-*training* dengan dataset dengan *preprocessing* Ben Graham.
- Model *nakhon* adalah model yang di-*training* dengan dataset dengan *preprocessing* Nakhon.
- Model *ramasubramanian* adalah model yang di-*training* dengan dataset dengan *preprocessing* Ramasubramanian.
- Model *enhanced_green* adalah model yang di-*training* dengan dataset dengan *preprocessing enhanced green*.

Demikian juga untuk *training* model *fine tuning n* blok Inception v3. Hasil *training* ini melihat akurasi yang diberikan oleh setiap model dengan jumlah blok yang di-*fine tuning* berbeda.

A. *Pengaruh Preprocessing - Training Classifier*

Hasil *training* bagian *classifier* dari lima kondisi dataset dapat dilihat pada Tabel VI. Nilai ini diukur dari dataset *validation* setelah *training* sebanyak 50 *epoch* selesai dan merupakan rata-rata dari 5-*fold cross validation*. Pada tabel tersebut dapat dilihat bahwa model *enhanced_green* memberikan hasil yang paling baik dengan nilai: akurasi 76.10%, *precision* 81.92%, *recall* 69.90% dan AUC 95.12%. Model kedua terbaik diberikan oleh model *ramasubramanian* dengan nilai akurasi 74.10%. Model *rescale_png* menempati posisi ketiga, sedangkan model *ben_graham* dan *nakhon* menempati posisi keempat dan kelima.

TABEL VI
METRIK HASIL *TRAINING* CLASSIFIER

Model	***Loss***	***Acc***	***Precision***	***Recall***	**AUC**
rescale_png	1.94	73.78%	80.70%	66.81%	94.53%
ben_graham	2.09	72.73%	79.52%	65.09%	94.16%
nakhon	2.02	69.10%	78.70%	60.09%	93.05%
ramasubramanian	1.91	74.10%	81.26%	67.41%	94.92%
enhanced_green	1.93	76.10%	81.92%	69.90%	95.12%

F1-*score* untuk setiap kelas DR dari lima model ini dapat dilihat pada Tabel VII. Pada tabel tersebut kelas 0 memberikan nilai paling tinggi. Hal ini dikarenakan kelas 0 memiliki jumlah citra paling banyak. Sedangkan kelas 3 memberikan nilai paling kecil karena kelas 3 memiliki jumlah citra paling sedikit meskipun pembobotan kelas selama *training* sudah diberikan. Dari tabel ini juga dapat dilihat bahwa F1-*score* tertinggi dari setiap kelas DR diberikan oleh model yang berbeda. DR kelas 0 diberikan oleh model *ramasubramanian*, DR kelas 1 diberikan oleh model *rescale_png*, DR kelas 2 dan 4 diberikan oleh model *enhanced_green*. DR kelas 3 diberikan oleh model *ramasburamanian* dan *enhanced_green*.

TABEL VII
F1-SCORE *TRAINING CLASSIFIER*

DR Level	***rescale_png***	***graham***	***nakhon***	***ramasubramanian***	***enhanced_green***
0	95.6%	95.2%	95.0%	**96.4%**	95.8%
1	**52.6%**	46.2%	46.2%	49.6%	46.2%
2	59.6%	56.2%	48.6%	57.6%	**64.6%**
3	36.6%	31.4%	34.4%	**37.6%**	**37.6%**
4	38.2%	42.2%	30.4%	41.4%	**47.4%**

B. *Pengaruh Preprocessing - Fine Tuning*

Setelah *training* bagian *classifier* selesai, *fine tuning* dilakukan sebanyak dua blok Inception v3. Hasil *fine tuning* dari setiap model dapat dilihat pada Tabel VIII. Nilai ini diukur dari dataset *validation* setelah *training* sebanyak 50 *epoch* selesai dan merupakan rata-rata dari 5-*fold cross validation*. Pada tabel tersebut tampak bahwa model *enhanced_green* memberikan hasil yang paling baik dengan nilai: akurasi 78.79%, *precision* 82.97%, *recall* 74.64%, dan AUC 95.81%. Model *rescale_png* menempati posisi kedua dengan nilai akurasi 77.27% yang diikut model *ramasubramanian* di posisi ketiga dengan nilai akurasi 76.73%. Sedangkan model *ben_graham* dan *nakhon* menempati posisi keempat dan kelima.

TABEL VIII
METRIK HASIL *FINE TUNING*

Model	***Loss***	***Acc***	***Precision***	***Recall***	**AUC**
rescale_png	1.83	77.27%	80.79%	72.41%	95.32%
ben_graham	1.98	75.59%	80.35%	70.04%	94.97%
nakhon	1.89	72.30%	79.56%	65.72%	94.25%
ramasubramanian	1.81	76.73%	81.17%	72.44%	95.62%
enhanced_green	1.83	78.79%	82.97%	74.64%	95.81%

F1-*score* untuk setiap kelas DR dari lima model ini dapat dilihat pada Tabel IX. Pada tabel tersebut kelas 0 memberikan nilai paling tinggi. Hal ini dikarenakan kelas 0 memiliki jumlah citra paling banyak. Sedangkan kelas 3 memberikan nilai paling kecil karena kelas 3 memiliki jumlah citra paling sedikit meskipun pembobotan kelas selama *training* sudah diberikan. Dari tabel ini juga dapat dilihat bahwa F1-*score* tertinggi dari setiap kelas DR diberikan oleh model yang berbeda. DR kelas 0, 1, dan 3 diberikan oleh model *ramasubramanian*. DR kelas 2 dan 4 diberikan oleh model *enhanced_green*.

TABEL IX
F1-SCORE *FINE TUNING*

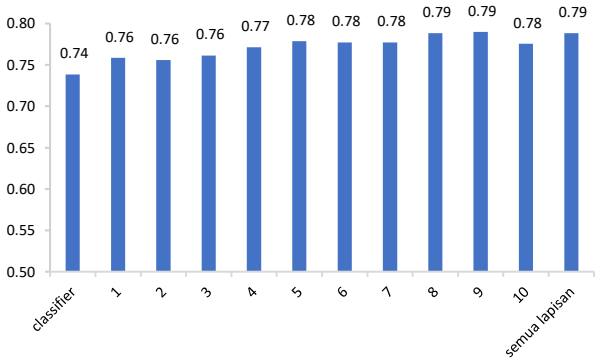
DR Level	***rescale_png***	***graham***	***nakhon***	***ramasubramanian***	***enhanced_green***
0	96.2%	96.6%	96.2%	**97.4%**	96.4%
1	55.2%	53.0%	51.0%	**55.4%**	52.2%
2	67.2%	60.6%	55.0%	62.4%	**68.8%**
3	41.4%	36.6%	35.6%	**42.0%**	41.4%
4	43.4%	44.8%	33.4%	44.8%	**52.0%**

C. *Fine Tuning n blok inception*

Hasil *training n* blok Inception v3 bisa dilihat pada Tabel X. Nilai pada tabel tersebut diukur pada dataset *validation* setelah *training* sebanyak 50 *epoch* selesai. Gambar 17 adalah visualisasi Tabel X dalam bentuk grafik batang untuk nilai akurasi. Sumbu *x* pada Gambar 17 adalah jumlah blok Inception yang di-*training* dan sumbu *y* adalah nilai akurasi model. Jika mengacu pada grafik ini, *fine tuning* memberikan hasil terbaik jika dilakukan sebanyak sembilan blok Inception. Tampak juga pada grafik bahwa *fine tuning* dari lapisan manapun memberikan *benefit* terhadap performa model dengan rentang kenaikan akurasi antara 1.71% sampai 5.14%.

TABEL X
PERFORMA MODEL YANG DI-*TRAINING* SEBANYAK *N* BLOK INCEPTION V3

Blok	***Loss***	***Acc***	***Precision***	***Recall***	**AUC**
classifier	1.95	73.86%	80.78%	67.86%	94.21%
1	1.86	75.86%	79.84%	69.57%	94.73%
2	1.86	75.57%	79.62%	72.00%	94.97%
3	1.84	76.14%	79.68%	71.71%	95.16%
4	1.84	77.14%	81.79%	73.14%	95.24%
5	1.82	77.86%	81.79%	74.43%	95.41%
6	1.82	77.71%	81.36%	73.57%	95.53%
7	1.81	77.71%	81.49%	74.86%	95.64%
8	1.81	78.86%	81.64%	75.57%	95.54%
9	1.81	79.00%	81.41%	74.43%	95.59%
10	1.82	77.57%	81.40%	75.00%	95.49%
Semua lapisan	1.81	78.86%	81.85%	76.00%	95.74%



Gambar 17. Akurasi model dengan *fine tuning n* blok Inception v3

V. SIMPULAN

Dalam penelitian ini didapati bahwa tidak semua *preprocessing* memberikan hasil yang baik. Dari empat *preprocessing* yang dilakukan, yaitu Graham, Nakhon, Ramasubramanian dan *enhanced green*, didapati bahwa *enhanced green* memberikan performa yang paling baik. Model *enhanced_green* memberikan nilai akurasi 76.10% pada *training classifier* dan nilai akurasi 78.79% setelah *fine tuning*. Selisih akurasi model antara *enhance green* dengan model tanpa *preprocessing* adalah 1.52%. *Preprocessing enhanced green* mengambil *channel* hijau dari citra *fundus* retina lalu dilakukan *image enhancement* dengan CLAHE untuk memperbaiki kontras citra dan diikuti dengan *unsharp masking* untuk menajamkan citra. Citra hasil *preprocessing enhanced green* disimpan sebagai citra tiga *channel* di mana setiap *channel*-nya adalah *channel* hijau.

Fine tuning n blok Inception yang dilakukan pada penelitian ini menunjukkan bahwa *fine tuning* bisa dilakukan mulai dari blok Inception v3 mana pun dan tetap memberikan *benefit* terhadap performa model. *Fine tuning* memberikan rentang kenaikan akurasi antara 1.71% sampai 5.14%. Jika dilihat dari hasil percobaan, *fine tuning* sebanyak sembilan blok Inception v3 memberikan hasil terbaik dengan nilai akurasi 79.0%.

Berdasarkan hasil temuan dari eksplorasi yang telah dilakukan, terdapat sejumlah eksplorasi lanjutan yang dapat dilakukan. Penelitian selanjutnya dapat melakukan *hyperparameter tuning* dan menggunakan beberapa teknik optimasi *training* model. *Hyperparameter* yang bisa di-*tuning* seperti *learning rate*, *ukuran batch*, nilai probabilitas *dropout*, dan nilai *weight decay*. Teknik optimasi yang bisa diimplementasikan saat *training* model seperti *learning rate decay*, *reduce learning rate on plateau*, ataupun *early stopping*. Arsitektur CNN yang dipilih bisa menggunakan arsitektur yang lebih baru seperti MobileNet, Inception v4, ataupun EfficientNet. Arsitektur *classifier* pun bisa dieksplorasi dengan menambahkan *hidden layer* atau pun *hidden unit*.

Penelitian berikutnya bisa menggunakan ukuran citra yang lebih besar, misalnya $598 \times 598 \times 3$. Memperbesar ukuran citra *training* diharapkan dapat memberikan akurasi model yang lebih baik. *Preprocessing* yang digunakan dalam penelitian ini juga bisa digunakan pada dataset lain untuk melihat *robustness* pengaruh setiap *preprocessing* terhadap performa model.

Kesulitan terbesar dalam penelitian ini adalah interpretasi model. Model dipandang sebagai *black box* yang menerima *input* dan mengeluarkan *output*. *Output* berupa performa model yang diukur pada penelitian ini. Jika penelitian dilakukan bersama dengan seorang *ophthalmologist* maka interpretasi model bisa lebih baik untuk melihat pengaruh *preprocessing* terhadap fitur pada citra *fundus* retina.

DAFTAR PUSTAKA

[1] M. Sazzad Hossen, A. A. Reza, dan M. C. Mishu, "An automated model using deep convolutional neural network for retinal image classification to detect diabetic retinopathy," dalam *ACM International Conference Proceeding Series*, 2020, hlm. 1–8, doi: 10.1145/3377049.3377067.

[2] N. Asiri, M. Hussain, F. Al Adel, dan N. Alzaidi, "Deep learning based computer-aided diagnosis systems for diabetic retinopathy: A

survey," *Artif. Intell. Med.*, vol. 99, no. July, hlm. 101701, 2019, doi: 10.1016/j.artmed.2019.07.009.

[3] Y. W. Chen, T. Y. Wu, W. H. Wong, dan C. Y. Lee, "Diabetic Retinopathy Detection Based on Deep Convolutional Neural Networks," *ICASSP IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2018-April, hlm. 1030–1034, 2018, doi: 10.1109/ICASSP.2018.8461427.

[4] A. Krizhevsky, I. Sutskever, dan G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, hlm. 84–90, Mei 2017, doi: 10.1145/3065386.

[5] N. Tajbakhsh *dkk.*, "Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, hlm. 1299–1312, Mei 2016, doi: 10.1109/TMI.2016.2535302.

[6] "ImageNet." [Online]. Tersedia pada: http://image-net.org/about-stats. [Diakses: 13-Okt-2020]

[7] J. Yosinski, J. Clune, Y. Bengio, dan H. Lipson, "How transferable are features in deep neural networks?," *Adv. Neural Inf. Process. Syst.*, vol. 4, no. January, hlm. 3320–3328, 2014.

[8] M. A. Bravo dan P. A. Arbelaez, "Automatic Diabetic Retinopathy Classification," hlm. 10.

[9] C. Lian, Y. Liang, R. Kang, dan Y. Xiang, "Deep convolutional neural networks for diabetic retinopathy classification," *ACM Int. Conf. Proceeding Ser.*, no. c, hlm. 68–72, Jun 2018, doi: 10.1145/3239576.3239589.

[10] S. Mohammadian, A. Karsaz, dan Y. M. Roshan, "Comparative Study of Fine-Tuning of Pre-Trained Convolutional Neural Networks for Diabetic Retinopathy Screening," dalam *2017 24th National and 2nd International Iranian Conference on Biomedical Engineering (ICBME)*, Tehran, 2017, hlm. 1–6, doi: 10.1109/ICBME.2017.8430269 [Online]. Tersedia pada: https://ieeexplore.ieee.org/document/8430269/. [Diakses: 25-Jun-2020]

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, dan Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," dalam *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, hlm. 2818–2826, doi: 10.1109/CVPR.2016.308 [Online]. Tersedia pada: http://ieeexplore.ieee.org/document/7780677/. [Diakses: 25-Jun-2020]

[12] Asia Pacific Tele-Ophthalmology Society, "APTOS 2019 Blindness Detection dataset," *APTOS 2019*, 2019. .

[13] I. Goodfellow, Y. Bengio, dan A. Courville, *Deep learning*. Cambridge, Massachusetts: The MIT Press, 2016.

[14] G. Currie, K. E. Hawk, E. Rohren, A. Vial, dan R. Klein, "Machine Learning and Deep Learning in Medical Imaging: Intelligent Imaging," *J. Med. Imaging Radiat. Sci.*, vol. 50, no. 4, hlm. 477–487, 2019, doi: 10.1016/j.jmir.2019.09.005.

[15] C. Szegedy *dkk.*, "Going deeper with convolutions," dalam *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07-12-June, no. 8, hlm. 1–9, doi: 10.1109/CVPR.2015.7298594.

[16] X. Wang, Y. Lu, Y. Wang, dan W. B. Chen, "Diabetic retinopathy stage classification using convolutional neural networks," *Proc. - 2018 IEEE 19th Int. Conf. Inf. Reuse Integr. Data Sci. IRI 2018*, hlm. 465–471, 2018, doi: 10.1109/IRI.2018.00074.

[17] S.-H. Tsang, "Review: Inception-v3 — 1st Runner Up (Image Classification) in ILSVRC 2015," *Medium*, 23-Mar-2019. [Online]. Tersedia pada: https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c. [Diakses: 11-Jan-2021]

[18] "vision.learner | fastai." [Online]. Tersedia pada: https://fastai1.fast.ai/vision.learner.html#cnn_learner. [Diakses: 04-Jan-2021]

[19] "Fitting Batch Norm into a neural network," *Coursera*. [Online]. Tersedia pada: https://www.coursera.org/learn/deep-neural-network/lecture/RN8bN/fitting-batch-norm-into-a-neural-network. [Diakses: 30-Des-2020]

[20] Z. Gao, J. Li, J. Guo, Y. Chen, Z. Yi, dan J. Zhong, "Diagnosis of Diabetic Retinopathy Using Deep Neural Networks," *IEEE Access*, vol. 7, hlm. 3360–3370, 2019, doi: 10.1109/ACCESS.2018.2888639.

[21] Dr. A. Rosebrock, *Deep Learning for Computer Vision with Python - Starter Bundle*, First printing., vol. 1, 3 vol. PYIMAGESEARCH, 2017.

[22] G. García, J. Gallardo, A. Mauricio, J. López, dan C. Del Carpio, "Detection of Diabetic Retinopathy Based on a Convolutional Neural Network Using Retinal Fundus Images," dalam *Artificial Neural Networks and Machine Learning – ICANN 2017*, vol. 10614, A. Lintas, S. Rovetta, P. F. M. J. Verschure, dan A. E. P. Villa, Ed. Cham: Springer International Publishing, 2017, hlm. 635–642 [Online]. Tersedia pada: http://link.springer.com/10.1007/978-3-319-68612-7_72. [Diakses: 05-Des-2020]

[23] M. T. Hagos dan S. Kant, "Transfer Learning based Detection of Diabetic Retinopathy from Small Dataset," 2019.

[24] "Image hashing with OpenCV and Python - PyImageSearch." [Online]. Tersedia pada: https://www.pyimagesearch.com/2017/11/27/image-hashing-opencv-python/. [Diakses: 31-Des-2020]

[25] B. Graham, "Kaggle Diabetic Retinopathy Detection competition report," hlm. 9.

[26] "APTOS : Eye Preprocessing in Diabetic Retinopathy." [Online]. Tersedia pada: https://kaggle.com/ratthachat/aptos-eye-preprocessing-in-diabetic-retinopathy. [Diakses: 19-Des-2020]

[27] B. Ramasubramanian dan S. Selvaperumal, "A comprehensive review on various preprocessing methods in detecting diabetic retinopathy," dalam *2016 International Conference on Communication and Signal Processing (ICCSP)*, Melmaruvathur, Tamilnadu, India, 2016, hlm. 0642–0646, doi: 10.1109/ICCSP.2016.7754220 [Online]. Tersedia pada: http://ieeexplore.ieee.org/document/7754220/. [Diakses: 05-Des-2020]

[28] Aravind Eye Care System, "4th Asia Pacific Tele-Ophthalmology Society (APTOS) Symposium," *APTOS 2019*, 2019. .

[29] H. Pratt, F. Coenen, D. M. Broadbent, S. P. Harding, dan Y. Zheng, "Convolutional Neural Networks for Diabetic Retinopathy," *Procedia Comput. Sci.*, vol. 90, no. July, hlm. 200–205, 2016, doi: 10.1016/j.procs.2016.07.014.

[30] "scikit-learn/scikit-learn," *GitHub*. [Online]. Tersedia pada: https://github.com/scikit-learn/scikit-learn. [Diakses: 02-Jan-2021]

[31] "python - Crop black border of image using NumPy," *Code Review Stack Exchange*. [Online]. Tersedia pada: https://codereview.stackexchange.com/questions/132914/crop-black-border-of-image-using-numpy. [Diakses: 13-Jan-2021]

[32] R. C. Gonzalez dan R. E. Woods, *Digital image processing*, 3. ed., internat. ed. Upper Saddle River, NJ: Pearson, 2010.

[33] "OpenCV: Operations on arrays." [Online]. Tersedia pada: https://docs.opencv.org/master/d2/de8/group__core__array.html#gafafb2513349db3bcff51f54ee5592a19. [Diakses: 12-Jan-2021]

[34] J. Sanjaya dan M. Ayub, "Augmentasi Data Pengenalan Citra Mobil Menggunakan Pendekatan Random Crop, Rotate, dan Mixup," *J. Tek. Inform. Dan Sist. Inf.*, vol. 6, no. 2, Agu 2020, doi: 10.28932/jutisi.v6i2.2688. [Online]. Tersedia pada: https://journal.maranatha.edu/index.php/jutisi/article/view/2688. [Diakses: 05-Des-2020]

[35] F. Chollet, *Deep learning with Python*. Shelter Island, New York: Manning Publications Co, 2018.

[36] "tf.keras.initializers.GlorotUniform | TensorFlow Core v2.4.0." [Online]. Tersedia pada: https://www.tensorflow.org/api_docs/python/tf/keras/initializers/GlorotUniform. [Diakses: 01-Jan-2021]

[37] "Advanced Guide to Inception v3 on Cloud TPU," *Google Cloud*. [Online]. Tersedia pada: https://cloud.google.com/tpu/docs/inception-v3-advanced. [Diakses: 11-Jan-2021]