# BAB 4 METODOLOGI PENELITIAN

Secara umum penelitian ini mengklasifikasikan citra *fundus* retina ke dalam lima kelas berdasarkan tingkatannya. Citra *fundus* retina akan diklasifikasikan ke dalam salah satu kelas: 0-normal, 1-DR ringan, 2-DR sedang, 3-DR parah, atau 4-DR proliferative. Penelitian ini mengimplementasi teknik *deep learning* CNN yang mengekstrak fitur secara otomatis dari sebuah citra untuk tugas klasifikasi. Arsitektur CNN yang digunakan adalah Inception v3 karena arsitektur ini yang paling banyak menjadi pilihan seperti yang sudah dipaparkan pada Tabel 3.10.

## 4.1 Hipotesis Penelitian

Sesuai dengan rumusan masalah dan tujuan penelitian, maka hipotesis yang diangkat pada penelitian ini adalah:

1. *Transfer learning* akan memberikan hasil yang lebih baik dibandingkan dengan *end to end learning* meskipun domain citra *pretrained* berbeda dengan domain citra *fundus* retina.
2. *Preprocessing* yang tepat akan memberikan dampak positif terhadap performa model. Dengan *preprocessing* diharapkan fitur yang diinginkan pada citra menjadi lebih menonjol.
3. *Fine tuning* hanya perlu dilakukan pada beberapa blok terkahir saja dari model yang di-*training* dengan pendekatan *transfer learning*. Pada lapisan awal, *fine tuning* tidak memberikan pengaruh terhadap performa model karena lapisan awal CNN mengekstrak fitur umum dari citra.

## 4.2 *Pipeline* Penelitian

*Pipeline* penelitian ini dapat dilihat pada Gambar 4.1. Penelitian diawali dengan studi literatur, menentukan rumusan masalah, tujuan penelitian, pemilihan metode, dan hipotesis yang diuji. Dataset citra *fundus* retina diambil dari Kaggle dengan judul kompetisi “APTOS 2019 Blindness Detection”. Eksperimen diawali dengan membuang duplikasi citra dalam dataset [38], lalu semua citra diubah ukurannya sehingga memiliki radius *pixel* 300 pada wilayah retina [13]. Citra yang

sudah diubah ukurannya ini dilanjutkan dengan empat *preprocessing* terpisah, yaitu:

1. *Preprocessing* Ben Graham [13].
2. *Preprocessing* Nakhon Ratchasima [14].
3. *Preprocessing* Ramasubramanian [15].
4. *Preprocessing enhanced green*.

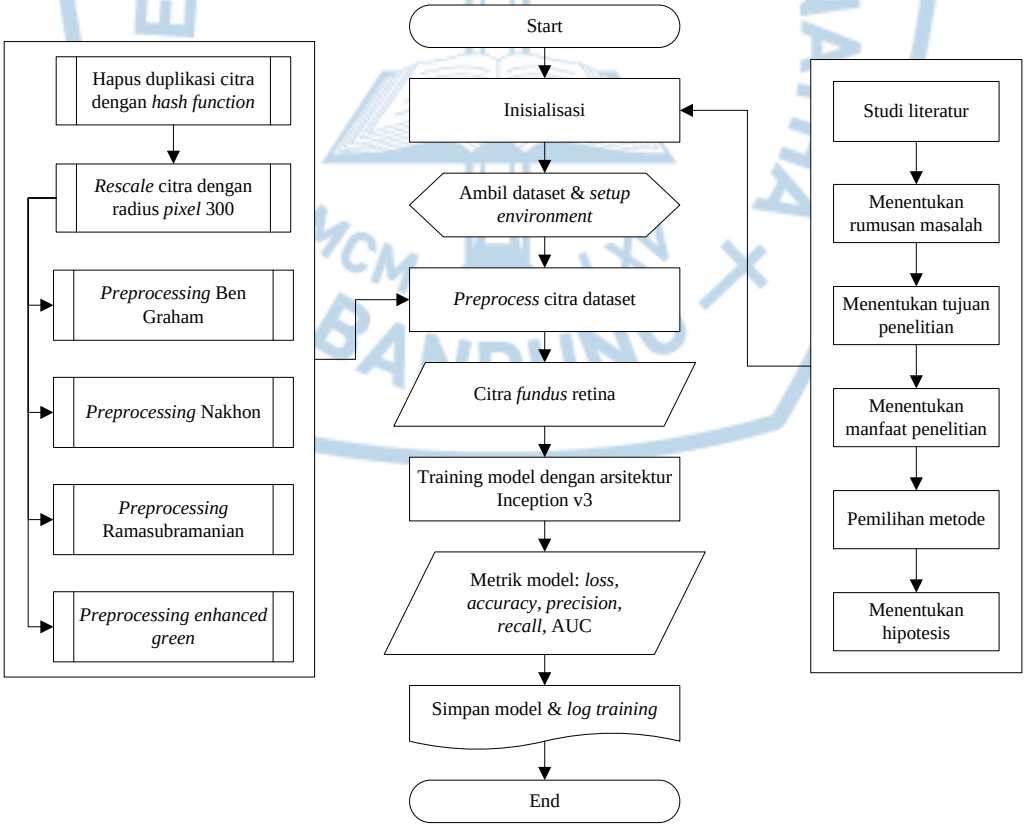
Rangkaian *Preprocessing* ini dapat dilihat hubungannya pada Gambar 4.3.

Setelah tahap *preprocessing* selesai, penelitian dilanjutkan dengan *training* model dengan menggunakan teknik *deep learning* CNN. *Training* model dilakukan dengan beberapa pendekatan dan beberapa kondisi dataset, yaitu:

1. Model yang di-*training* dengan pendekatan *end to end learning* dan menggunakan dataset citra yang diubah ukurannya.
2. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra yang diubah ukurannya. *Fine tuning* dilakukan sebanyak dua blok Inception v3.
3. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra dengan *preprocessing* yang diusulkan oleh Ben Graham [13]. *Fine tuning* dilakukan sebanyak dua blok Inception v3.
4. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra dengan *preprocessing* yang diusulkan oleh Nakhon Ratchasima [14]. *Fine tuning* dilakukan sebanyak dua blok Inception v3.
5. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra dengan *preprocessing* yang diusulkan oleh Ramasubramanian [15]. *Fine tuning* dilakukan sebanyak dua blok Inception v3.
6. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra dengan *preprocessing image enhancement* pada *channel* hijau dan disimpan sebagai citra tiga *channel* (*Green, Green, Green* / GGG); di mana setiap *channel*-nya merupakan *channel* hijau. *Fine tuning* dilakukan sebanyak dua blok Inception v3. *Fine tuning* dilakukan sebanyak dua blok Inception v3.

7. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra dengan *preprocessing image enhancement* pada *channel* hijau dan disimpan sebagai citra tiga *channel* (*Red*, *Green*, *Blue* / RGB); di mana *channel* R dan B nilainya adalah 0.
8. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra *file format* JPEG yang diubah ukurannya. Citra pada dataset ini memiliki *file format* JPEG kualitas 72. Selain model ini, model lainnya di-*training* dengan *file format* PNG. *Fine tuning* dilakukan sebanyak dua blok Inception v3.
9. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra yang diubah ukurannya. *Fine tuning* dilakukan sebanyak *n*-blok Inception v3, dengan $n$ = [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 yang dipilih sebagai metrik utama adalah akurasi.




**Gambar 4.1 Diagram alir penelitian**

## 4.3 **Dataset**

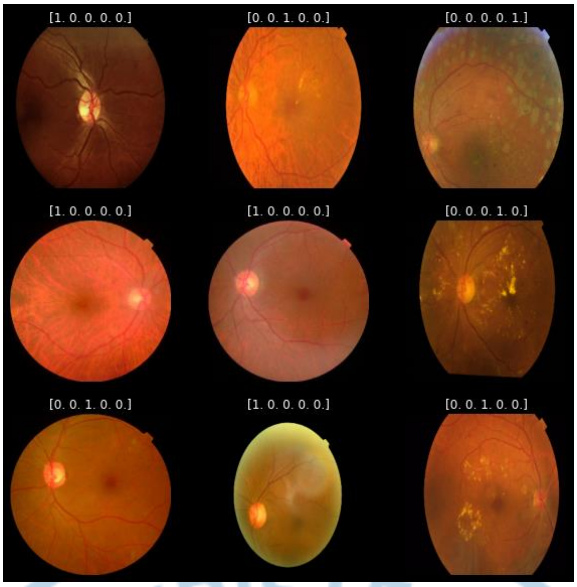
Dataset yang digunakan untuk *training* model diambil dari dari Kaggle.com [12]. Dataset ini berasal dari rumah sakit mata Aravind Eye Hospital [13] yang dikeluarkan bersamaan dengan kegiatan APTOS Symposium ke-4 [39]. Dataset APTOS 2019 ini diberikan kepada publik dengan tujuan untuk meningkatkan kemampuan rumah sakit mengidentifikasi tingkat keparahan 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 tingkat keparahan DR terhadap setiap citra dengan skala 0 – 4 oleh ahli klinis seperti yang dapat dilihat pada Tabel 4.1.

**Tabel 4.1 Label untuk citra sesuai dengan tingkat DR**

| Label | Label *one hot* | Indikasi | Jumlah Citra |
|---|---|---|---|
| 0 | [**1**, 0, 0, 0, 0] | Normal | 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 4.2. Pada gambar tersebut dapat dilihat citra *fundus* retina dengan labelnya yang *encoded* dalam bentuk *one hot vector*. Dataset yang diberikan oleh Kaggle terdiri dari:

1. train.csv - nama citra dengan labelnya
2. test.csv - kumpulan nama citra yang akan diidentifikasi
3. sample_submission.csv - contoh format yang harus dikumpulkan untuk kompetisi Kaggle
4. train.zip - kumpulan citra *training*
5. test.zip - kumpulan citra *test*




**Gambar 4.2 Contoh citra *fundus* retina beserta dengan labelnya.**

Sedangkan deskripsi citra dalam dataset adalah sebagai berikut:

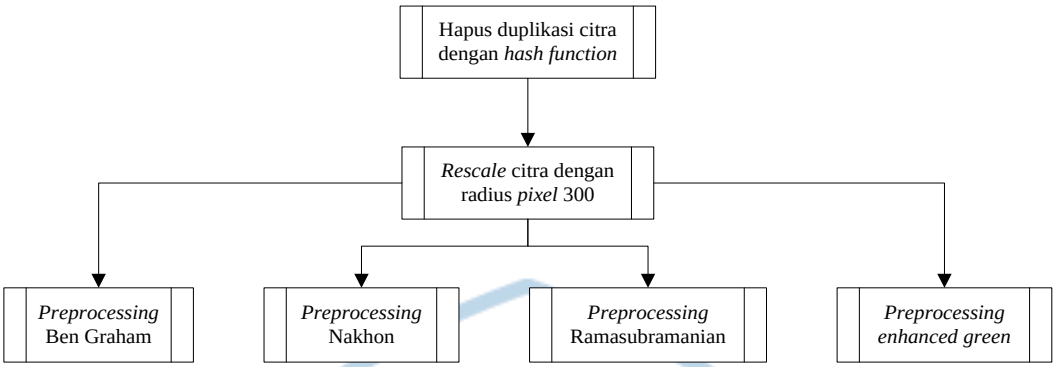
1. File format citra: PNG
2. Variasi dimensi citra: $474 \times 358\ pixel$ – $3.388 \times 2.588\ pixel$
3. Variasi ukuran file citra: 217 kB – 7.495 kB
4. Jumlah citra *training*: 3.662 citra
5. Jumlah citra *test*: 1.928 citra

Dataset APTOS tidak menyediakan label sebagai dasar kebenaran untuk citra *test*, sehingga pada penelitian kali ini menggunakan citra *training* yang dibagi menjadi citra *training* dan citra test dengan teknik 5-*fold cross validation*.

## 4.4 *Preprocessing* Dataset

Jika dilihat pada Tabel 3.10, *preprocessing* yang umum dilakukan pada citra *fundus* retina merujuk pada *preprocessing* yang dilakukan oleh Ben Graham. Selain *preprocessing* tersebut terdapat *preprocessing* yang ditawarkan oleh Nakhon Ratchasima [14], yang merupakan *notebook preprocessing* dengan *vote* terbanyak pada kompetisi Kaggle dengan judul "APTOS 2019 Blindness Detection". Ramasubramanian [15] juga menawarkan *preprocessing* yang 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. Adapun rangkaian *preprocessing* dapat dilihat pada Gambar 4.3




**Gambar 4.3 Diagram alir *preprocessing***

### 4.4.1 Menghapus Duplikasi Citra

Sebelum masuk ke tahap *preprocessing* citra, dataset diperiksa terlebih dahulu, apakah terdapat duplikasi citra. Algoritma *difference hash* (*dhash*) digunakan untuk memberikan *signature* atau identitas pada setiap gambar [38]. Dari *signature* tersebut dapat ditemukan duplikasi jika terdapat dua citra atau lebih dengan signature yang sama. *Signature* diperoleh dengan mengambil $n$ digit bilangan biner dari sebuah citra. Adapun langkah untuk membuat nilai *dhash* adalah:

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

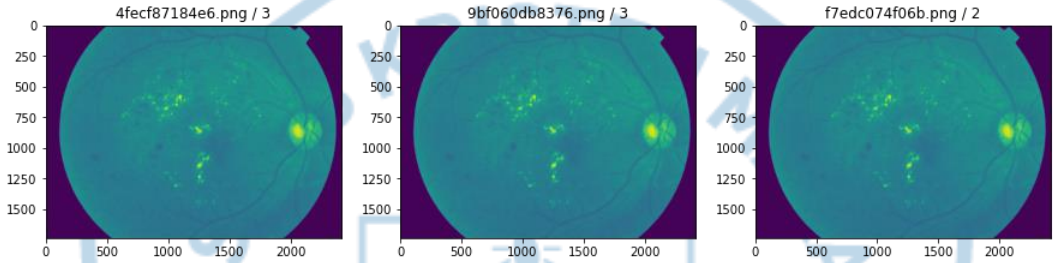
$$signature = \sum_{i=0}^{n-1} 2^{\mathrm{i}} \tag{4.1}$$

Nilai $n$ yang dipilih 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:

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

Pada Gambar 4.4 dapat dilihat duplikasi citra sebanyak tiga buah dengan label yang berbeda. Citra dengan kondisi seperti ini dieliminasi ketiganya dari dataset. Jumlah citra setelah duplikasi dieliminasi dari dataset dapat dilihat pada Tabel 4.2.




**Gambar 4.4. Contoh gambar duplikasi dengan label yang berbeda**

**Tabel 4.2. 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 | |

### 4.4.2 **Mengubah Ukuran Citra**

Teknik mengubah ukuran citra mengadopsi teknik yang dilakukan Ben Graham [13]. Citra diskalakan ulang 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`

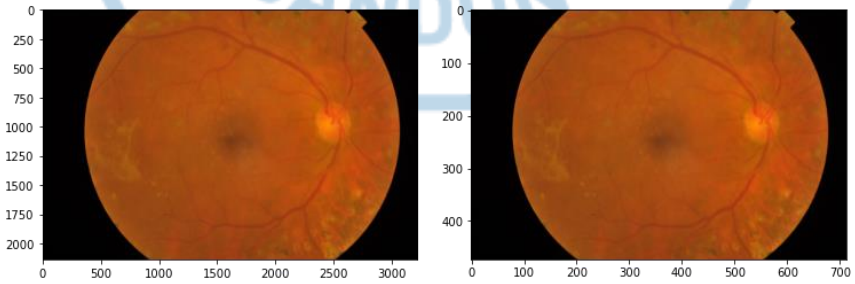
**Output**: `s` (faktor skala citra)

```
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` tersebut dengan radius *pixel* pilihan yaitu 300 *pixel*. Nilai skalar `s` digunakan sebagai faktor skala untuk mengubah ukuran citra.

Teknik mengubah ukuran citra seperti ini tetap memperhatikan aspek rasio dari setiap citra. Hasil dari citra yang sudah diubah ukurannya dapat dilihat pada Gambar 4.5. Pada gambar tersebut dapat dilihat ukuran citra sebelum dan sesudah perubahan ukuran. Citra yang diubah ukurannya ini menjadi dasar untuk setiap *preprocessing* berikutnya karena citra yang ukurannya lebih kecil mengurangi beban komputasi.




**Gambar 4.5 Citra sebelum dan sesudah ukuran diubah**

### 4.4.3 *Preprocessing* Ben Graham

Terdapat tiga tahapan *preprocessing* yang dilakukan oleh Ben Graham [13], yaitu:

1. Mengubah ukuran citra dengan metode *rescale* yang sudah dibahas pada bagian 4.4.2.
2. Mengurangi warna citra dengan warna rata-rata lokal, lalu dipetakan ke 50% abu-abu
3. Memotong citra secara melingkar menjadi 90% dari ukuran asli untuk menghilangkan "*boundary effects*."

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

$$y(n,m) = \lambda \times x(n,m) + (-\lambda) \times g(n,m) + 128 \quad (4.2)$$

Di mana:

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

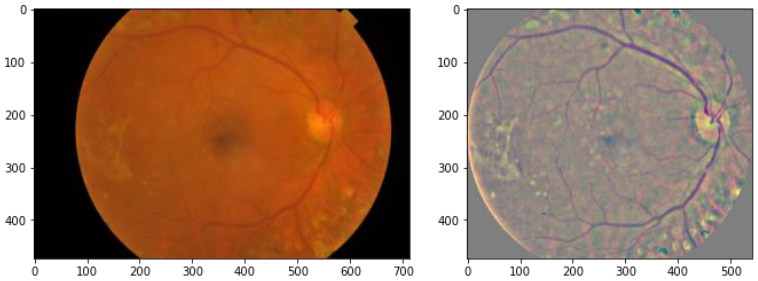
Nilai $g(n,m)$ diperoleh dengan persamaan:

$$g(n,m) = G(n,m,\sigma) * x(n,m) \quad (4.3)$$

Di mana:

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

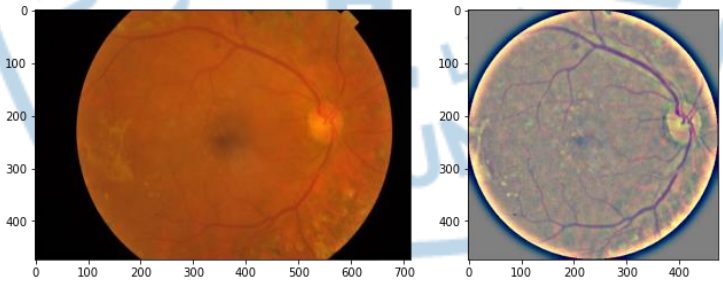
Hasil dari *preprocessing* Ben Graham dapat dilihat pada Gambar 4.6. Gambar sebelah kiri merupakan gambar *input*, yang belum melewati *preprocessing* Ben Graham dan gambar sebelah kanan adalah citra *output* yang sudah melewati *preprocessing* Ben Graham.



**Gambar 4.6 *Preprocessing* Ben Graham**

### 4.4.4 *Preprocessing* Nakhon

*Preprocessing* Nakhon [14] 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 [40]. Setelah margin hitam dibuang, proses selanjutnya serupa dengan *preprocessing* yang dilakukan oleh Ben Graham (proses 2), namun Nakhon tidak membuang "*boundary effect*." Nakhon memotong setiap citra secara melingkar, sehingga citra *fundus* retina memiliki bentuk yang sama. Gambar 4.7 menunjukan citra *input* dan citra *output* dari *preprocessing* Nakhon. Bisa dilihat pada citra *output*, bentuk retina dipotong dengan melingkar sempurna.

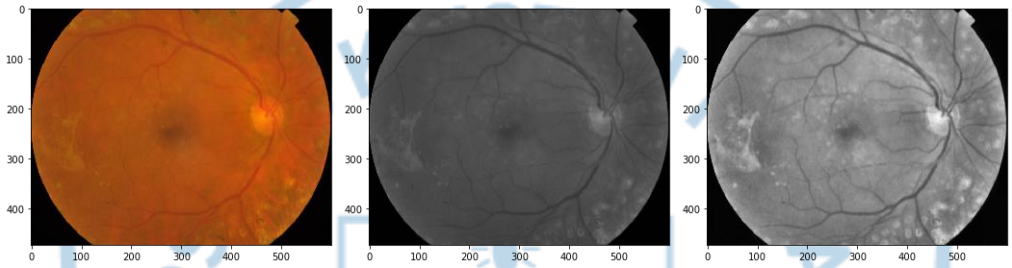


**Gambar 4.7 *Preprocessing* Nakhon Ratchasima**

### 4.4.5 *Preprocessing* Ramasubramanian

Ramasubramanian dan Selvaperumal [15] mengusulkan teknik *preprocessing* dengan mengambil *green channel* dari citra RGB karena *green channel* memiliki informasi paling banyak dan menunjukan diskriminasi antara gejala klinis dengan warna latar belakang. Mula-mula citra diambil *channel* hijau lalu margin hitam dipotong. Median *filter* digunakan untuk menghilangkan *salt and*

*pepper noise* sambil mempertahankan garis tepi pada citra fundus retina. Ukuran *kernel* median filter yang digunakan adalah $3 \times 3$. Setelah *noise* dihilangkan, citra ditingkatkan dengan *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,* dan *blood vessel* akan lebih terlihat. Hasil *preprocessing* usulan Ramasubramanian dapat dilihat pada Gambar 4.8. Dari kiri ke kanan: citra sebelum *prerprocessing*, citra *green channel*, citra, citra yang sudah melewati *preprocessing* Ramasubramanian dan dilipatkan menjadi tiga *channel*.



**Gambar 4.8 Preprocessing Ramasubramanian dan Selvaperumal**

### 4.4.6 ***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 $8 \times 8$ lalu *histogram equalization* diterapkan pada wilayah tersebut. Menurut Ramasubramanian [15], CLAHE akan menonjolkan fitur tersembunyi seperti *exudates, microaneurysms, fovea,* dan *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 [41]:

1. Citra asli diburamkan. Teknik untuk memburamkan citra menggunakan *Gaussian blur* seperti yang diusulkan oleh Ben Graham.

2. Kurangi citra asli dengan citra yang sudah diburamkan. Proses ini akan membentuk *mask*.
3. Tambahkan citra asli dengan *mask* yang terbentuk

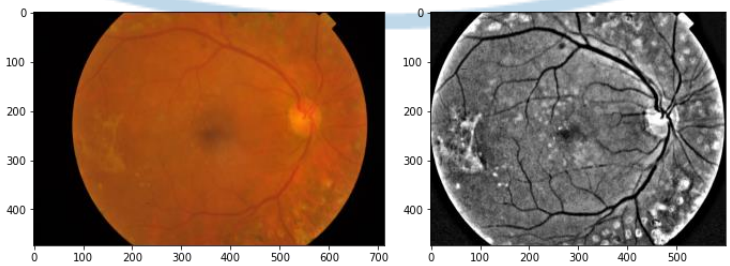
*Unsharp masking* bisa diberikan dengan persamaan [42]:

$$y(n,m) = \alpha \times x(n,m) + \beta \times g(n,m) \quad (4.4)$$

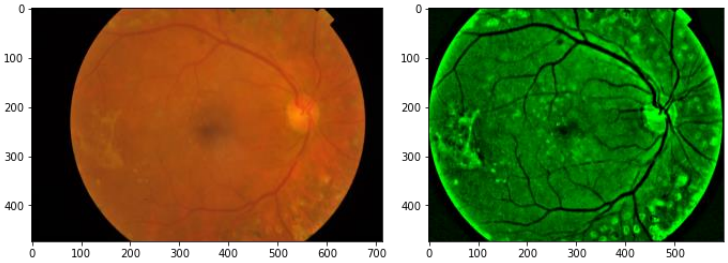
Di mana:

- $y(n,m)$ adalah citra *output.*
- $x(n,m)$ adalah citra *input.*
- $g(n,m)$ adalah citra yang diburamkan dengan *Gaussian blur*. $g(n,m)$ diperoleh dari persamaan 3.3
- $\alpha$ adalah koefisien citra asli. $\alpha$ ditetapkan dengan nilai 4 mengikuti koefisien yang dipilih oleh Ben Graham.
- $\beta$ adalah $1 - \alpha$.

Hasil *preprocessing enhanced green* dapat dilihat pada Gambar 4.9 dan Gambar 4.10. Citra sebelah kiri merupakan citra *input*. Citra sebelah kanan merupakan citra *output* dari *preprocessing enhanced green*. Pada Gambar 4.9 adalah citra tiga *channel* di mana ketiga *channel*-nya adalah *channel* hijau, sedangkan pada Gambar 4.10 adalah citra tiga *channel* di mana nilai *channel* R dan B adalah 0. Dapat dilihat pada gambar tersebut, objek pada citra berupa titik atau garis lebih tajam.




**Gambar 4.9 Preprocessing Enhanced Green (G, G, G)**



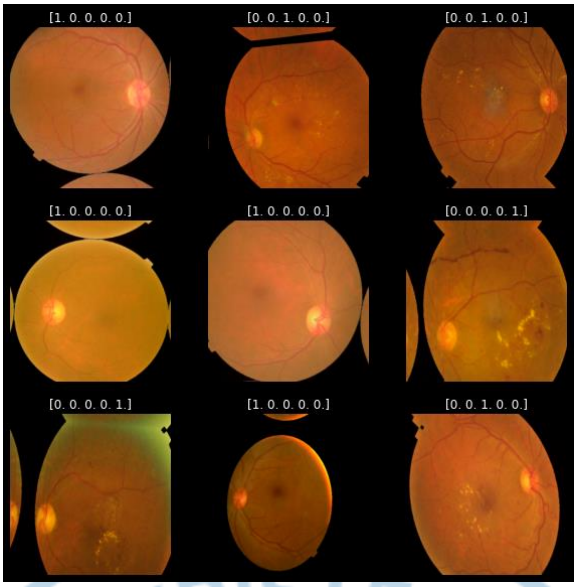
**Gambar 4.10 Preprocessing Enhanced Green (0,G,0)**

## 4.5 **Augmentasi**

Augmentasi citra adalah strategi yang memungkinkan praktisi untuk meningkatkan keragaman data yang tersedia untuk *training* model, tanpa benar-benar mengumpulkan data baru [43]. 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 [44]. Citra yang diaugmentasi dapat dilihat pada Gambar 4.11. Augmentasi yang dipakai adalah:

1. Rotasi citra: 10°
2. Pergeseran citra pada sumbu $x$ dan sumbu $y$
3. Distorsi citra dengan *shear* sebesar 0.1
4. *Zoom*: 90% – 110%
5. *Flipping* horizontal dan vertikal

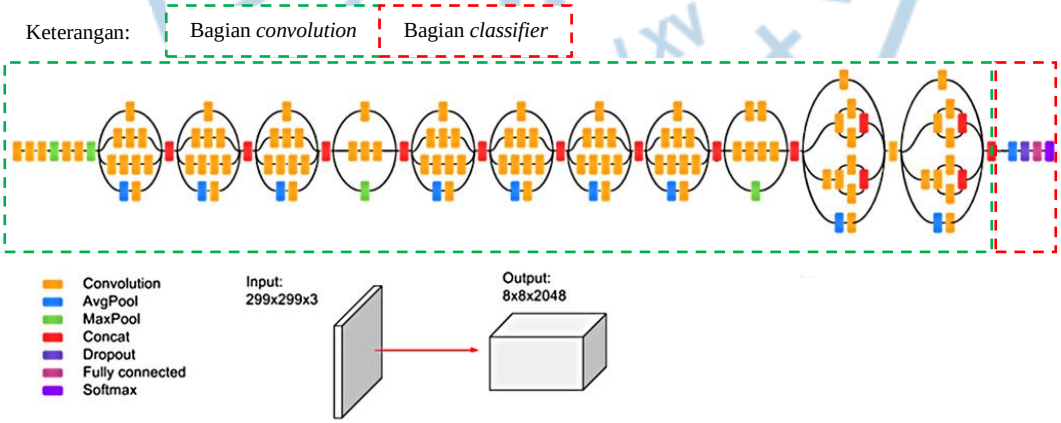
Augmentasi ini hanya diterapkan untuk citra *training* saja, sedangkan untuk citra *test* tidak dilakukan augmentasi.




**Gambar 4.11 Citra *training* dengan augmentasi**

## 4.6 *Training* Model

Model di-*training* menggunakan *library* Keras, yang merupakan *high level* API dari *platform deep learning* Tensorflow. Pada Keras terdapat berbagai model yang sudah di-*training* dengan dataset ImageNet, termasuk model dengan arsitektur Inception v3. Dengan memanfaatkan *library* Keras, arsitektur Inception v3 bisa dipanggil sebagai sebuah fungsi, tidak perlu didefinisikan lagi lapisan demi lapisan. Versi *library* yang digunakan bisa dilihat pada Lampiran A.




**Gambar 4.12 Dua bagian utama Arsitektur** Inception v3

Secara umum arsitektur Inception v3 dapat dibagi menjadi dua bagian utama, yaitu bagian *convolution* dan bagian *classifier*. Bagian *convolution* adalah kumpulan dari lapisan *convolution*, *pooling*, dan aktivasi ReLU yang bertugas

untuk mengekstrak fitur dari citra *input*. Bagian *classifier* tersusun atas *Neural Network* yang bertugas untuk mengklasifikasikan fitur yang sudah diekstrak oleh bagian *convolution*. *Training* model dijalankan dengan empat skema utama yang menghasilkan delapan model, yaitu:

1. *Training* model dengan pendekatan *end to end learning*. Bobot model diinisialisasi secara acak dengan Glorot Uniform dan dataset yang dipakai pada *training* ini adalah dataset asli, tanpa *preprocessing*, hanya diubah ukurannya saja.
2. *Training model* untuk melihat pengaruh *preprocessing* terhadap performa model. *Training* model dilakukan dengan pendekatan *transfer learning* dan diikuti *fine tuning* sebanyak dua blok Inception v3 sesuai dengan percobaan yang sudah dilakukan oleh Saboora [10]. Dataset yang dipakai pada *training* ini adalah:
    a. Dataset yang diubah ukurannya.
    b. Dataset yang dilakukan *preprocessing* Ben Graham
    c. Dataset yang dilakukan *preprocessing* Nakhon Ratchasima
    d. Dataset yang dilakukan *preprocessing* Ramasubramanian
    e. Dataset yang dilakukan *preprocessing enhanced green* dan disimpan sebagai tiga *channel* G, G, G.
    f. Dataset yang dilakukan *preprocessing enhanced green* dan disimpan sebagai tiga *channel* R, G, B; di mana nilai *channel* R dan B adalah 0.
3. *Training* model untuk melihat banyaknya blok Incpetion v3 yang perlu dilakukan *fine tuning* pada pendekatan *transfer learning. Fine tuning* dilakukan terhadap *n* blok Inception v3, mulai dari *n* = [1, 2, …, 10] dan semua lapisan. Dataset yang dipakai pada *training* ini adalah dataset asli yang diubah ukurannya.
4. *Training* model untuk melihat perbedaan *file format* dan kualitas gambar terhadap performa model yang dihasilkan. Dataset yang digunakan adalah dataset yang diubah ukurannya dan dikonversi menjadi JPEG dengan kualitas 72.

Agar setiap model dapat dibandingkan, maka beberapa *hyperparameter* tidak diubah untuk setiap *training* delapan model tersebut. Model di-*training* dengan citra berukuran $299 \times 299 \times 3$ karena model *pre-trained* di-*training* dengan ukuran $299 \times 299 \times 3$ pada dataset ImageNet. *Batch size* ditetapkan sebesar 32 sesuai dengan kemampuan *hardware*. Optimasi yang dipilih adalah ADAM karena penelitian Saboora [10] menunjukan optimasi ADAM lebih baik jika dibandingkan dengan optimasi SGDM. *Loss function* yang dipilih adalah *categorical cross entropy* karena model mengklasifikasikan citra *input* menjadi lima kelas. Nilai *weight decay* menggunakan nilai awal yang disediakan oleh *library* Keras yaitu 0.01. Total *epoch* yang dipilih untuk setiap *training* model adalah 100 *epoch*. Pada *training* model *transfer learning*, 100 *epoch* akan dibagi dua, 50 *epoch* untuk *training classifier* dan 50 *epoch* untuk *fine tuning*. Setelah 100 *epoch* selesai, model akan dievaluasi dengan empat metrik pengukuran. Metrik yang diukur dari performa model adalah akurasi, *precision*, *recall*, dan AUC. model dibatasi *Hyperparameter* yang dipakai untuk semua model dapat dilihat pada Tabel 4.3. *Hyperparameter* yang berlaku khusus disampaikan pada masing-masing sub-bab sesuai dengan *training* model yang dilakukan (subbab 4.6.1 – subbab 4.6.4)

**Tabel 4.3 Hyperparameter umum yang dipakai setiap *training* model**

| No | Hyperparameter | Nilai |
|---|---|---|
| 1 | Ukuran citra | 299 x 299 x 3 |
| 2 | *Batch Size* | 32 |
| 3 | Optimasi | ADAM |
| 4 | *Loss function* | *Categorical cross entropy* |
| 5 | Metrik | Akurasi, *precision*, *recall*, AUC |

Pada Tabel 4.2 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 [31] dan Gabriel Garcia [34] menggunakan pembobotan terhadap 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 persamaan [45]:

$$w_j = \frac{\sum_j n_j}{c \times n_j} \tag{4.5}$$

Di mana:

- $w_j$ 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 4.4.

**Tabel 4.4 Bobot dari setiap kelas DR**

| Tingkat DR | Setelah | Bobot kelas |
|---|---|---|
| 0 | 1796 | 0.389532 |
| 1 | 338 | 2.069822 |
| 2 | 921 | 0.759609 |
| 3 | 173 | 4.043931 |
| 4 | 270 | 2.591111 |
| **Total** | **3.498** | |

Dengan pemberian bobot terhadap setiap kelas, maka nilai *loss function categorical crossentropy* pada persamaan 2.7 menjadi:

$$L_i = -w_j \times log\left(\frac{e^{s_{yi}}}{\sum_j e^{s_j}}\right) \tag{4.6}$$

Di mana:

- $L_i$ adalah nilai loss citra ke-$i$
- $s_{yi}$ adalah nilai *output* citra $i$ untuk kelas yang sesuai dengan *ground truth*
- $s_j$ adalah nilai *output* citra $i$ untuk semua kelas yang mungkin

Nilai *output* atau *scoring function* diperoleh dengan persamaan 2.5 [16].

Inisialisasi Glorot Uniform digunakan jika terdapat bagian dari arsitektur atau model yang perlu diinisialisasi secara acak. Pada model dengan *training end to end learning*, keseluruhan bobot baik bagian *classifier* maupun *convolution* diinisialisasi dengan Glorot Uniform. Sedangkan bobot pada model dengan *training*

*transfer learning*, hanya bagian *classifier* saja yang diinisialisasi dengan Glorot Uniform. Bobot pada bagian *convolution* menggunakan bobot yang sudah di-*training* dari dataset ImageNet. Inisialisasi Glorot Uniform mengambil sampel nilai dari distribusi seragam antara $[-limit, limit]$ [46].

$$limit = \sqrt{\frac{6}{fan_{in} + fan_{out}}} \tag{4.7}$$

Di mana:

- $fan_{in}$ adalah jumlah *input* unit
- $fan_{out}$ adalah jumlah *output* unit.

### 4.6.1 **Pemilihan Arsitektur *Classifier***

Sebelum memulai *training* model timbul satu pertanyaan, arsitektur *classifier* seperti apa yang cocok untuk ditambahkan pada bagian akhir dari CNN. Pertanyaan ini dijawab dengan melakukan percobaan kecil untuk memilih arsitektur *classifier*. Percobaan kecil ini tidak menggunakan keseluruhan dataset, hanya menggunakan 750 citra *training* dan 115 citra uji yang dipilih secara acak dan dibuat seimbang untuk setiap kelasnya. Terdapat tiga arsitektur *classifier* yang diuji dan *classifier* yang memberikan tingkat akurasi terbaik akan digunakan untuk penelitian selanjutnya terhadap keseluruhan dataset.

1. Arsitektur pertama adalah arsitektur *classifier* berdasarkan arsitektur asli Inception v3 yang digunakan oleh Christian Szegedy, Zhentao Gao, dan *default classifier* pada library Keras [11], [27], [30]. Arsitektur *classifier* ini diberi nama arsitektur Keras.
2. Arsitektur kedua adalah arsitektur *classifier* yang digunakan sebagai *classifier default* untuk *transfer learning* pada *library* Fastai [47]. Arsitektur ini diberi nama arsitektur *fastai*.
3. Arsitektur ketiga adalah modifikasi dari arsitektur fastai. Arsitektur ini mengubah posisi lapisan *Batch Normalization* sebelum fungsi aktivasi ReLU. Hal ini dilakukan berdasarkan materi pembelajaran yang

disampaikan oleh Andrew Ng [48]. Arsitektur ini diberi nama *fastai modified*.

Tiga arsitektur tersebut dapat dilihat pada Gambar 4.13, Gambar 4.14, dan Gambar 4.15. Pada gambar-gambar tersebut dapat dilihat setiap lapisan dengan dimensi *tensor* input dan output. Misalkan dimensi tensor pada InputLayer tertulis `[(?, 299, 299, 3)]` artinya adalah:

- ? adalah nilai *batch size* yang ditentukan sebesar 32
- 299 adalah jumlah baris *pixel* citra input
- 299 adalah jumlah kolom *pixel* citra input
- 3 adalah jumlah *channel* citra input

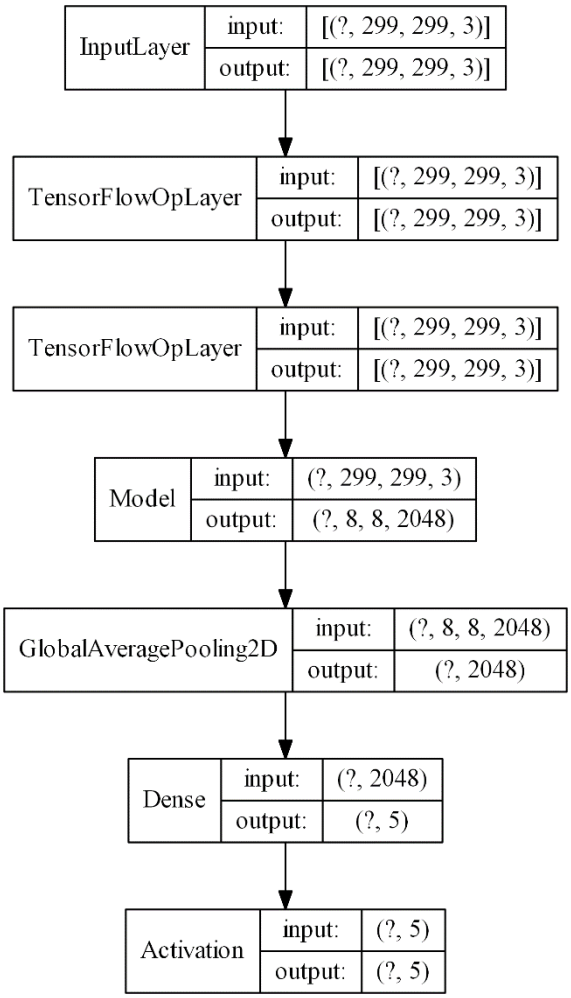
Setiap arsitektur di-*training* sebanyak 50 *epoch* dengan *learning rate* $1e^{-4}$. *Hyperparameter* yang digunakan untuk memilih *classifier* bisa dilihat pada Tabel 4.5.

**Tabel 4.5 Hyperparameter yang digunakan selama *training* untuk memilih *classifier***

| No | Hyperparameter | Nilai |
|---|---|---|
| 1 | *Learning rate* | $1e^{-4}$ |
| 2 | *Epoch* | 50 |

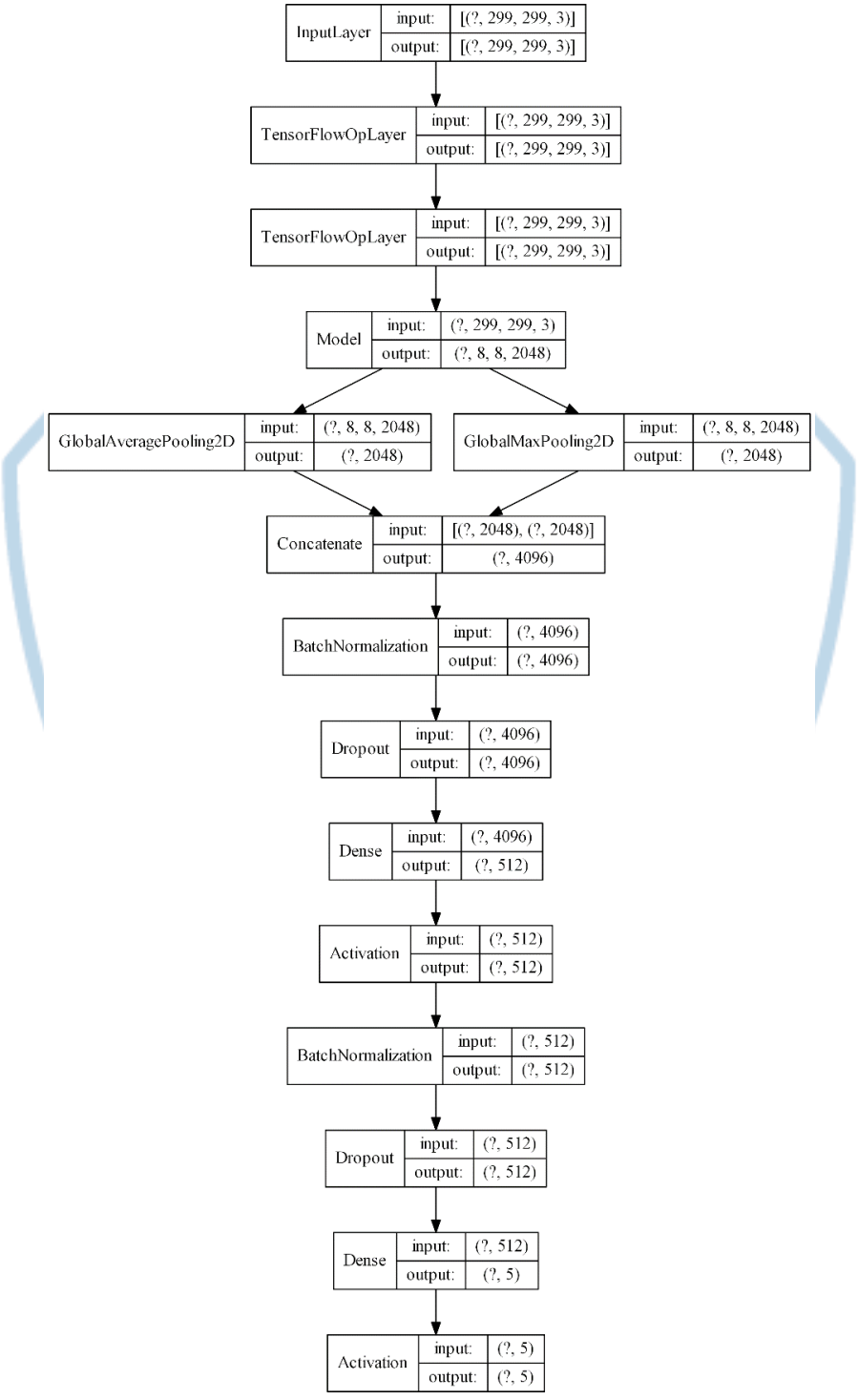
Berikut ini ketiga arsitektur yang diuji:

1. Arsitektur Keras tersusun atas lapisan: *Global Average Pooling* dan *output*. Berikut ini diagram arsitektur *classifier* yang dimaksud:



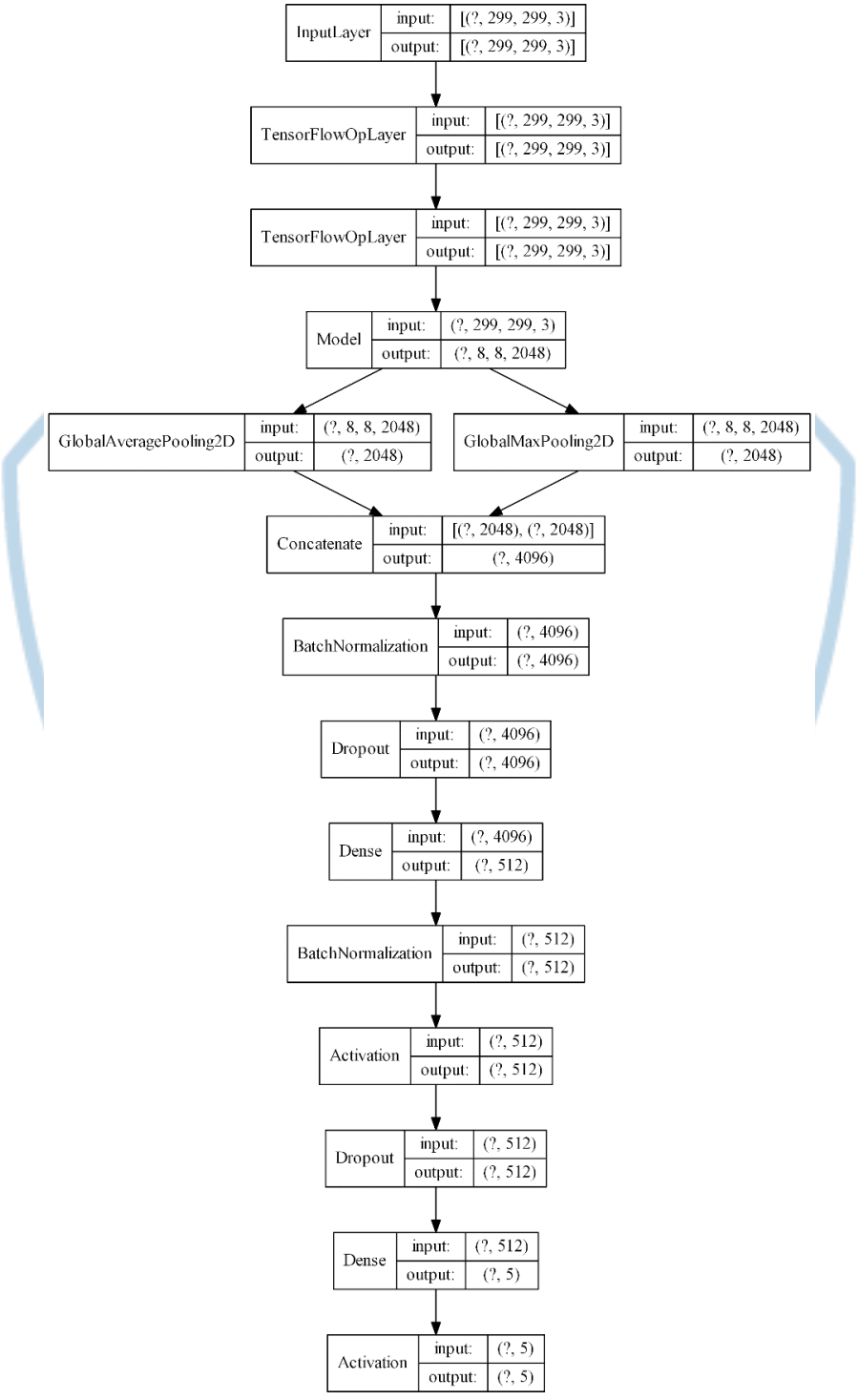

**Gambar 4.13 Arsitektur *classifier* yang dipakai oleh Zhentao Gao dan default pada *library* Keras**

2. Arsitektur *fastai* tersusun atas lapisan: *Global Average Pooling* dan *Global Max Pooling* yang digabungkan, *Batch Normalization*, *dropout*, *fully connected* (512 unit), aktivasi ReLU, *batch normalization*, *dropout*, dan *output*. Berikut ini diagram arsitektur yang dimaksud.




**Gambar 4.14 Arsitektur *classifier* dari *library* Fastai**

3. Arsitektur *fastai modified* tersusun atas lapisan: *global average pooling* dan *global max pooling* yang digabungkan, *batch normalization*, *dropout*, *fully connected* (512 unit), *batch normalization*, aktivasi ReLU, *dropout*, dan *output*.




**Gambar 4.15 Arsitektur *classifier* yang dimodifikasi dari Fastai**

### 4.6.2 *End to End Learning*

Bobot model *training end to end learning* diinisalisasi dengan Glorot Uniform, baik pada bagian *convolution* maupun *classifier*. Traninig model dilakukan sebanyak 100 *epoch* dengan *learning rate* $1e^{-4}$ yang bisa dilihat pada Tabel 4.6. Selama *training*, bobot dari setiap lapisan akan diperbaharui. Dataset dibagi menjadi 80% untuk *training* dan 20% untuk *test* dengan metode 5-*fold cross validation*. Nilai akhir akurasi model dihitung dari rata-rata 5-*fold cross validation*.

**Tabel 4.6 Hyperparameter khusus *training* model *end to end learning***

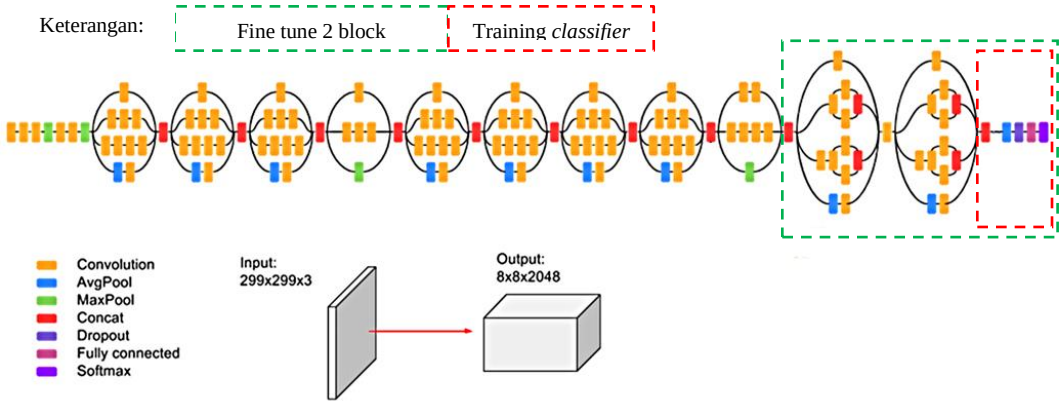
| No | Hyperparameter | Nilai |
|---|---|---|
| 1 | *Learning rate* | $1e^{-4}$ |
| 2 | *Epoch* | 100 |
| 3 | *Weight decay* | 0.01 |

Training model *end to end learning* akan menghasilkan satu buah model, yaitu:

1. Model yang di-training dengan pendekatan end to end learning dan menggunakan dataset citra yang diubah ukurannya.

### 4.6.3 *Transfer Learning* – Pengujian Pengaruh *Preprocessing*

Trainin model ini dilakukan dengan pendekatan *transfer learning*. Bagian *convolution* pada model *transfer learning* menggunakan bobot yang sudah di-*training* sebelumnya dari dataset ImageNet. Sedangkan bobot bagian *classifier* yang ditambahkan pada bagian akhir CNN diinisalisasi dengan Glorot Uniform. *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* $2e^{-6}$. Ilustrasi dari *training* ini dapat dilihat pada Gambar 4.16 dan *hyperparameter* yang digunakan dapat dilihat pada Tabel 4.7.




**Gambar 4.16 *Training* model dengan pendekatan *transfer learning***

Dataset dibagi menjadi 80% untuk *training* dan 20% untuk *test* dengan metode *5-fold cross validation*. Nilai akhir akurasi model dihitung dari rata-rata 5-*fold cross validation*.

**Tabel 4.7 *Hyperparameter* khsusus *training* model *transfer learning***

| No | *Hyperparameter* | Nilai |
|---|---|---|
| 1 | *Learning rate classifier* | $1e^{-4}$ |
| 2 | *Epoch classifier* | 50 |
| 3 | *Learning rate tuning* | $2e^{-6}$ |
| 4 | *Epoch fine tuning* | 50 |
| 5 | *Weight decay* | 0.01 |

Training model *transfer learning* akan menghasilkan tujuh model, yaitu:

1. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra yang diubah ukurannya. *Fine tuning* dilakukan sebanyak dua blok Inception v3.
2. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra dengan *preprocessing* yang diusulkan oleh Ben Graham [13]. *Fine tuning* dilakukan sebanyak dua blok Inception v3.
3. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra dengan *preprocessing* yang diusulkan oleh Nakhon Ratchasima [14]. *Fine tuning* dilakukan sebanyak dua blok Inception v3.
4. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra dengan *preprocessing* yang diusulkan oleh

Ramasubramanian [15]. *Fine tuning* dilakukan sebanyak dua blok Inception v3.

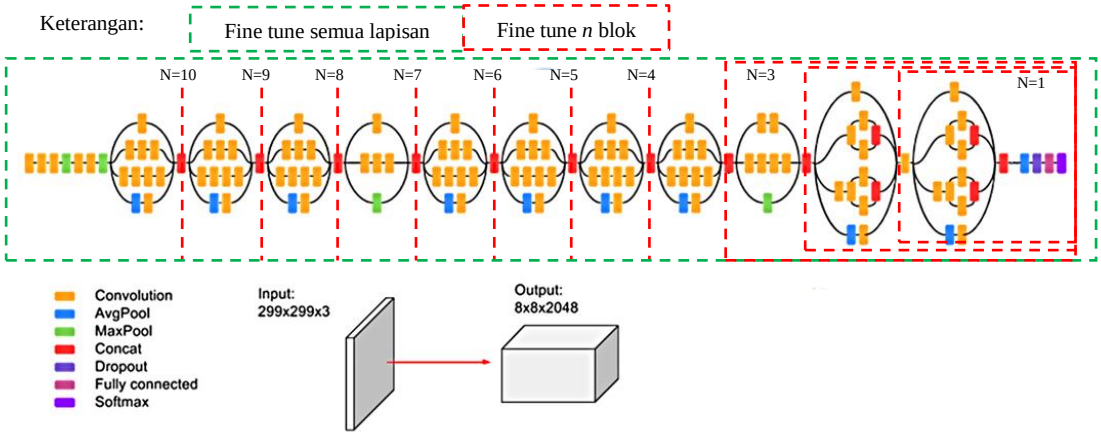
5. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra dengan *preprocessing enhanced green*. Citra yang digunakan adalah citra tiga *channel* (*Green*, *Green*, *Green* / GGG); di mana setiap *channel*-nya merupakan *channel* hijau. *Fine tuning* dilakukan sebanyak dua blok Inception v3.
6. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra dengan *preprocessing enhanced green*. Citra yang digunakan adalah citra tiga *channel* (*Red*, *Green*, *Blue* / RGB); di mana nilai *channel* R dan B adalah 0. *Fine tuning* dilakukan sebanyak dua blok Inception v3.
7. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra *file format* JPEG yang diubah ukurannya. Citra pada dataset ini memiliki *file format* JPEG kualitas 72. Selain model ini, model lainnya di-*training* dengan *file format* PNG. *Fine tuning* dilakukan sebanyak dua blok Inception v3.

### 4.6.4 ***Transfer Learning – Fine Tuning n* Blok Inception v3**

*Training* model ini menggunakan dataset yang diubah ukurannya. Bagian *convolution* pada model ini menggunakan bobot yang sudah di-*training* sebelumnya dari dataset ImageNet. Sedangkan bagian *classifier* yang ditambahkan pada bagian akhir CNN diinisalisasi dengan Glorot Uniform. Mula-mula *training* model dilakukan pada bagian *classifier*, lalu diikuti dengan *fine tuning* sebanyak *n* blok, $n = [1,2,3, \dots ,10]$ dan semua lapisan. *Training* bagian *classifier* dilakukan sebanyak 50 *epoch* dengan *learning rate* $1e^{-4}$. Selama *training classifier*, bobot pada bagian *convolution* dibekukan sehingga tidak diperbaharui. *Fine tuning* terhadap *n* blok Inception dilakukan sebanyak 50 *epoch* dengan *learning rate* $2e^{-6}$. *Fine tuning* terhadap *n* blok Inception diilustrasikan oleh Gambar 4.17. *Hyperparameter training* dapat dilihat pada Tabel 4.8. Dataset dibagi ke dalam dua bagian, yaitu 80% train dan 20% test dengan teknik *holdout*, bukan dengan 5-*fold cross validation*.

**Tabel 4.8 *Hyperparameter* khusus untuk *fine tuning n* blok Inception**

| No | Hyperparameter | Nilai |
|---|---|---|
| 1 | *Learning rate classifier* | 1e-4 |
| 2 | *Epoch classifier* | 50 |
| 3 | *Learning rate* tuning | 2e-6 |
| 4 | *Epoch* tuning | 50 |
| 5 | *Weight decay* | 0.01 |




**Gambar 4.17 Fine Tune n-block Inception v3 [38]**

Pada bagian ini akan menghasilkan satu buah model, yaitu:

1. Model yang di-*training* dengan pendekatan *transfer learning* dan menggunakan dataset citra yang diubah ukurannya. *Fine tuning* dilakukan sebanyak *n*-blok Inception v3, dengan $n$ = [1, 2, …, 10] dan semua lapisan.