Analisis *Optimizer* pada *Convolutional Neural Network* untuk Meningkatkan Akurasi PengenalanWajah

Hanjaya Suryalim[#1], Ken Ratri Retno Wardani[#2], Herry Heryanto[#3]

Departemen Informatika[#1],Departemen Informatika[#2],Departemen Informatika[#3]

ITHB[#1],ITHB[#2],ITHB[#3]

Jl. Dipati Ukur No. 80, Dago, Bandung, Indonesia

hanjayasuryalim@gmail.com
ken_ratri@ithb.ac.id
herry_heryanto@ithb.ac.id

Abstract*—The usage of face recognition system is arising day by day, that's why it is important to find a way to optimize accuracy rate of the model. Face recognition used CNN as its core technology. CNN consists of multiple layer of convolutional layers followed by fully connected layer. These convolutional layers have a primary function to extract important features from the image where those features will be used in fully connected layer to classify classes. This experiment will take 2 architecture called VGG16 and Inception to test acquired accuracy value using 2 dataset, Komnet and Yale dataset. This experiment also testing the effect of MCLAHE preprocessing on model accuracy. Accuracy of the model also depends on hyperparameter. This experiment will only test on optimizer and learning rate. Optimizer works by updating weight and bias value when backpropagation. Each optimizer has their own unique algorithm. The highest accuracy value in this experiment acquired by the model which trained using komnet dataset and using architecture Inception and adadelta as optimizer. Model acquired accuracy value of 98% in training process. Model also tested using 10 fold cross validation, and its average accuracy is 99.3%.

***Keywords*—face recognition, CNN, fully connected layer, MCLAHE, optimizer, learning rate, 10 fold cross validation, komnet, inception, VGG16.**

Abstrak*—Penggunaan sistem pengenalan wajah semakin meningkat dewasa ini, karena itu penting untuk menemukan cara yang optimal dalam meningkatkan akurasi pengenalan wajah. Pengenalan wajah memanfaatkan teknologi *CNN* yang tersusun dari lapisan-lapisan konvolusi yang diikuti oleh *fully connected layer*. Lapisan konvolusi ini yang bertanggungjawab atas proses ekstraksi fitur pada gambar, yang nantinya digunakan oleh *fully connected layer* untuk mengklasifikasi gambar tersebut. Pada penelitian kali ini diuji dua jenis arsitektur *CNN* yaitu VGG16 dan *Inception* untuk menguji tingkat akurasi yang dihasilkan. Pengujian juga menggunakan 2 jenis dataset yaitu Komnet dan *Yale*, serta menguji pengaruh *preprocessing MCLAHE* terhadap hasil akurasi. Selain jenis arsitektur, faktor lain seperti *hyperparameter* juga memegang andil tingkat akurasi model. *Hyperparameter*, yang diuji pada penelitian kali ini adalah jenis *optimizer* dan pengaruh perubahan *learning rate* pada penelitian. *Optimizer* bekerja dengan cara mengubah nilai bobot dan *bias* saat proses *backpropagation* dengan tujuan menghasilkan nilai *error* yang *minimum*. Setiap *optimizer* menggunakan algoritma yang unik. Hasil akurasi tertinggi yang dicapai adalah arsitektur *Inception* dengan *optimizer Adadelta* pada dataset Komnet. Hasil akurasi pada tahap pelatihan mencapai 98%. Rata-rata akurasi setelah model diuji dengan *10 fold cross validation* adalah 99.3%.

***Kata Kunci*—CNN, konvolusi, *fully connected layer*, *MCLAHE*, *hyperparameter*, *10 fold cross validation*, *optimizer*, *learning rate*, *optimizer*, *Inception*, VGG16.**

I. PENDAHULUAN

Pendeteksian wajah adalah teknik komputasi berbasis *artificial intelligence* yang digunakan untuk mendeteksi keberadaan wajah dalam suatu gambar digital. Dalam sistem pengenalan wajah, pendeteksian wajah digunakan untuk menentukan piksel-piksel yang akan diproses untuk sistem pengklasifikasian wajah. Pengklasifikasian wajah dalam sistem pengenalan wajah dilakukan dengan berbagai macam pendekatan. Dari beberapa sumber jurnal penelitian, dapat ditarik kesimpulan bahwa sistem pengenalan wajah dengan menggunakan pendekatan *deep learning* dan *Convolutional Neural Network (CNN)* memberikan hasil yang lebih optimal dibandingkan dengan metode *machine learning* konvensional [1].

Pada kasus pengujian terhadap dataset *Yale*, penggunaan *CNN* dengan arsitektur *Inception* mampu memberikan akurasi sebesar 99.89%. Dataset *Yale* merupakan data gambar *frontal face* dengan kondisi pencahayaan yang beragam. Pada kasus pengujian, data gambar yang digunakan terlebih dahulu diproses dengan metode *Modified Contrast Limited Adaptive Histogram Equalization* [1]. Tujuan dari pemrosesan ini adalah perataan *contrast* pada gambar. Penggunaan *MCLAHE* dengan *Inception* terbukti memberikan hasil yang optimal pada pengujian terhadap dataset *Yale*.

Metode optimasi digunakan untuk meningkatkan performa dari *machine learning* maupun *deep learning* [2]. Metode-metode yang dilakukan pada pemrosesan sistem *machine learning* pada dasarnya adalah fungsi optimasi. Tujuan dari fungsi optimasi adalah untuk mencari sekumpulan data masukan yang menghasilkan nilai keluaran baik itu minimum atau maksimum dari suatu fungsi objektif. Dalam sudut pandang *deep learning* maupun *machine learning*, fungsi objektif yang dimaksud adalah *cost function*. Serangkaian fungsi

optimasi yang diaplikasikan pada sistem *machine learning* ditujukan untuk mencapai nilai minimum dari *cost function*.

Performa dari *CNN* bergantung pada banyak faktor seperti *weight initialization, optimizer, batches, epochs, learning rate, activation function, loss function, network topology (architecture)*, kualitas data masukan dan kombinasinya [3]. Berbicara persoalan *segmentation* ataupun *classification*, pengujian dengan menggunakan satu *optimizer* tergolong pengujian yang bersifat lemah kecuali jika memiliki argumentasi yang cukup kuat menggunakan *optimizer* tertentu. Karenanya pemilihan *optimizer* merupakan proses yang penting. *Optimizer* berfungsi untuk mengurangi nilai *error* dengan cara mengubah nilai bobot dan *bias* pada *neural network*. Setiap *optimizer* memiliki algoritmanya sendiri dalam mengubah nilai bobot dan *bias*. Berdasarkan hasil pengujian penggunaan *optimizer* tidak mutlak selalu menghasilkan hasil yang lebih baik. Perlu dilakukan pengujian terhadap penggunaan *optimizer* itu sendiri terhadap suatu dataset [3].

Jurnal [3] menguji 10 jenis *optimizer* yaitu *Adaptive Gradient (Adagrad), Adaptive Delta (AdaDelta), Stochastic Gradient Descent (SGD), Adaptive Momentum (Adam), Cyclic Learning Rate (CLR), Adaptive Max Pooling (Adamax), Root Mean Square Propagation (RMSProp), Nesterov Adaptive Momentum (Nadam),* dan *Nesterov Accelerated Gradient (NAG)*. Diperoleh hasil terbaik adalah *Adam* dengan tingkat akurasi 99.2%.

Penelitian yang dilakukan oleh *Samar et al.* [4] mengembangkan arsitektur *CNN* 15 lapis dengan menggunakan *optimizer SGD* dan mencapai akurasi senilai 99.67 % dengan dataset *faces96*. Penelitian yang dilakukan oleh *Pranav* [5] meneliti pengaruh *hyperparameter tuning* pada arsitektur *CNN* sehingga menghasilkan akurasi sebesar 98.75 %.

Penelitian berikutnya yang dilakukan oleh *Musab et al.* [6] meneliti tentang pengaruh penambahan *batch normalization* terhadap akurasi pengenalan wajah. *Batch normalization* membantu memitigasi dampak dari *internal covariate shift* yang terjadi pada data gambar yang mengalami pemrosesan oleh bobot pada *hidden layer*. Akurasi tertinggi yang dicapai dalam penelitian ini adalah 94.8 %. Penelitian berikutnya yang dilakukan oleh *Zhiming Xie* [7] membahas tentang pengaruh perubahan jumlah *neuron* pada *convolution layer* terhadap kenaikan akurasi. Penelitian oleh *Xie* menggunakan arsitektur *LeNet-5*, diperoleh akurasi tertinggi dengan penggunaan jumlah neuron pada lapisan konvolusi masing-masing berjumlah 36,76 dan 1024.

Berdasarkan jurnal-jurnal referensi tersebut, penelitian ini akan menggunakan *MCLAHE* sebagai pemrosesan data awal. Seperti yang telah dijelaskan sebelumnya, *MCLAHE* adalah teknik perataan *contrast* pada gambar. Penelitian ini bermaksud menguji penggunaan *MCLAHE* terhadap akurasi model. Selanjutnya, untuk ekstraksi fitur digunakan *Inception* dan *VGG16*. Pemilihan arsitektur ini berdasarkan pengujian pada jurnal [1] yang merupakan jurnal utama penelitian ini.

Deep learning bekerja dengan dua proses yaitu *feed-forward propagation* dan *backward propagation*. *Feed-forward propagation* mempelajari fitur-fitur pada gambar dan menghasilkan model yang dipakai untuk memprediksi gambar. Hasil prediksi pada iterasi pertama memiliki kemungkinan tidak tepat dengan nilai yang seharusnya. Hal ini menyebabkan adanya nilai *error* pada sistem pelatihan. Tujuan dari pelatihan ini adalah untuk meminimalkan nilai *error* dari model dengan mengubah nilai bobot pada *hidden layer*. Nilai bobot pada *hidden layer* diubah pada saat *deep learning* melakukan proses *backward propagation*. Metode yang dipakai untuk merubah nilai bobot pada *hidden layer* ditentukan oleh jenis *optimizer* yang digunakan.

Penelitian kali ini bermaksud menguji 5 jenis *optimizer*(*Stochastic Gradient Descent, Nesterov Accelerated Gradient Descent, Adagrad, Adadelta & Adam*), untuk melihat dan meneliti *optimizer* yang menghasilkan tingkat akurasi paling tinggi. Selain itu penelitian ini juga menguji pengaruh perubahan *learning rate* terhadap tingkat akurasi. Penelitian ini juga membandingkan penggunaan arsitektur *Inception* dan *VGG16* terhadap nilai akurasi. Juga melihat pengaruh pengkoreksian gambar dengan metode *MCLAHE* terhadap akurasi.

II. METODOLOGI

A. Convolutional Neural Network

Convolutional Neural Network (CNN) atau yang biasa disebut juga *ConvNets* diperkenalkan pada tahun 1980 oleh Yann Lecun. Versi awal dari *CNN* dikenal dengan nama *LeNet*, digunakan untuk mengenali angka numerik yang ditulis oleh tangan. *CNN* terdiri dari lapisan bertingkat dari *neuron* buatan. *Neuron* buatan adalah imitasi kasar dari saraf / *neuron* secara biologis, yang pada dasarnya adalah fungsi matematis untuk menghitung jumlah bobot dari sejumlah nilai masukan dan hasil keluarannya dikenakan fungsi aktivasi [13].

CNN adalah pengembangan dari *multilayer perceptron* yang berupa *neural network* khusus untuk memproses data yang memiliki topologi seperti *grid*. Contoh data yang diproses adalah *time-domain* data yang dianggap sebagai *grid* satu dimensi yang mengambil sampel pada interval waktu dan data gambar yang dianggap sebagai *grid* piksel dua dimensi. Nama *convolutional neural network* menunjukkan bahwa jaringan menggunakan operasi matematika yang disebut konvolusi, yang merupakan jenis operasi linear khusus. Jaringan konvolusional adalah *neural network* yang menggunakan operasi konvolusi sebagai ganti dari penerapan matriks umum pada setidaknya satu dari lapisannya [33].

Dalam bentuk yang paling umum, konvolusi adalah operasi pada dua fungsi argumen yang bernilai *real*. Ada dua buah argumen yang dibutuhkan untuk melakukan operasi konvolusi: masukan dan *kernel*. Keluaran hasil konvolusi disebut sebagai *feature map*. Dalam aplikasi pembelajaran mesin, masukan berupa larik multidimensi dan *kernel* biasanya berupa larik multidimensi dari parameter yang diadaptasi oleh algoritma pembelajaran. Larik multidimensi ini disebut juga sebagai *tensor*. Setiap elemen masukan dan *kernel* harus disimpan secara terpisah. *Kernel* belajar dengan memperbaharui nilainya pada tahap pelatihan, yaitu *backward propagation*. Ukuran *kernel* ditentukan oleh peneliti. Pemilihan ukuran *kernel* bergantung kepada fitur yang ingin dipelajari dari masukan [33]. Terdapat tiga jenis lapisan yang membentuk

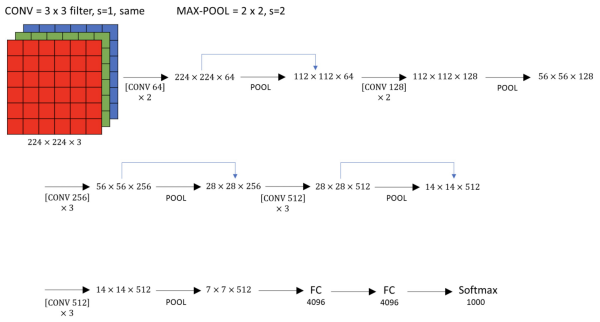

Fig. 1. Struktur VGG16

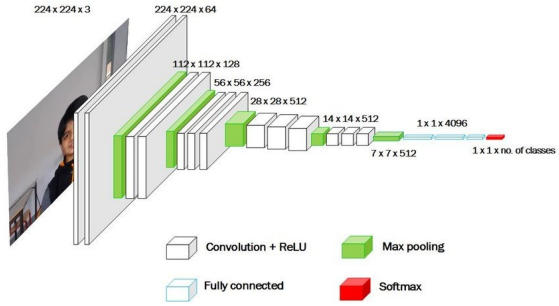

Fig. 2. Lapisan VGG16

arsitektur *convolutional neural network* yaitu: *convolutional layer*, *max pooling layer*, dan *dense/fully connected layer*.

1) VGG16

Ide dasar dari *VGG16* adalah menggunakan *network* yang lebih sederhana dibandingan dengan pendahulunya *(AlexNet)*. Yaitu dengan menggunakan lapisan konvolusi 3×3 dengan *stride* sebesar satu dan juga *same padding*. Di akhir setiap proses konvolusi, digunakan *max pooling* dengan nilai sama dengan 2 dan *stride* sebesar 2 [19].

Dimensi masukan data pada arsitektur *VGG16* adalah $224 \times 224 \times 3$. Gambar masukan kemudian diteruskan kepada lapisan konvolusi dengan filter 64 sebanyak 2 kali. Selanjutnya diteruskan kepada lapisan konvolusi dengan filter 128 sebanyak 2 kali. Selanjutnya, kembali diteruskan kepada lapisan konvolusi dengan filter 256 sebanyak 3 kali. Selanjutnya diteruskan kembali kepada lapisan konvolusi dengan filter 512 sebanyak 3 kali. Lalu dilanjutkan kepada lapisan konvolusi dengan filter 512 sebanyak 3 kali. Sebelum pada akhirnya dijadikan larik 1 dimensi dan diteruskan kepada *fully connected layer* / *dense layer* dengan filter 4096 sebanyak 2 kali. Lalu dilanjutkan kepada lapisan keluaran dengan fungsi aktivasi *softmax*.

Data masukan pada sistem memiliki tiga *channel*, lalu data masukan tersebut diteruskan kepada lapisan konvolusi dengan 64 *kernel* dan dimensi 3×3. yang terjadi disini adalah *kernel* yang sebenarnya dilewati adalah $64 \times 3 \times 3 \times 3$. [38]. Pada iterasi pertama terdapat tiga *kernel* yang dilewati pada tiga *channel* gambar. Hasil konvolusi masing-masing *channel* dijumlahkan menjadi nilai baru pada *channel* keluaran. Begitu seterusnya, sehingga hasil akhirnya adalah 64 *channel*.

Pada akhir setiap blok konvolusi, dapat ditambahkan lapisan *dropout* ataupun lapisan *batch normalization* sebagai metode regularisasi. Berdasarkan pembahasan oleh *Dr. Adrian Rosebrock* pada buku [24], pada penelitian ini diputuskan untuk mengaplikasikan *batch normalization* dan *dropout* pada arsitektur *VGG16*. Perhatikan figure 1 dan 2.

2) Inception

Ide utama dari arsitektur *inception* adalah, penggunaan lebih dari satu dimensi filter dalam satu kali konvolusinya. Filter ini berjalan secara paralel dan *feature map* hasil dari setiap

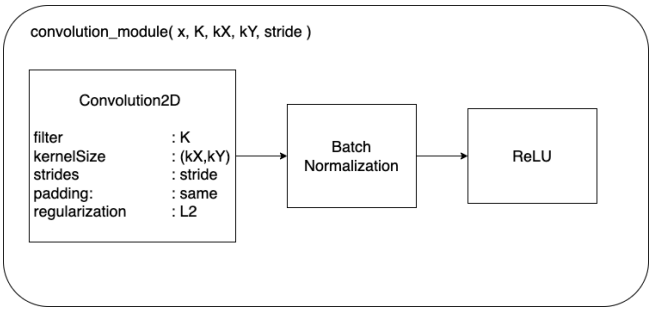

Fig. 3. *Convolution module*

filter konvolusi digabungkan menjadi satu kesatuan *feature map* [20].

Pada arsitektur *Inception* terdapat dua jenis *module* (kumpulan beberapa lapisan yang dipakai berulang), yaitu *convolution module* dan *inception module*. Pada figure 3 digambarkan ilustrasi *convolution module*. Berupa sebuah fungsi yang menerima parameter berupa banyaknya filter, ukuran filter dan besar nilai *strides*. Dalam *convolution module* terdapat proses konvolusi dua dimensi dengan *same padding* yang dilanjutkan oleh *batch normalization* lalu diakhiri oleh fungsi aktivasi berupa *ReLU*. Keluaran dari fungsi ini tentunya adalah *feature map*. Inisialisasi nilai bobot pada *kernel* di lapisan konvolusi menggunakan teknik regularisasi *L2*. Beberapa *convolution module* membentuk suatu *inception module*. Dalam arsitektur *Inception* terdapat tujuh *inception module* dan tiga *convolution module* yang dipakai. Proses penggabungan *feature map* berlangsung pada *inception module* [20].

Pada figure 4 dijelaskan bahwa digunakan konvolusi berdimensi 1×1. Fungsi dari konvolusi ini adalah untuk mengurangi jumlah parameter yang digunakan saat melakukan ekstraksi fitur. Penggunaan 1×1 *convolution* membantu mengurangi biaya komputasi. Teknik ini banyak dipakai oleh arsitektur yang rumit seperti *Squeeze Net*, *ResNet* dan *Inception*. Figure 5 menggambarkan struktur arsitektur *inception* secara keseluruhan.

B. Optimizer

Deep Learning merupakan *approximation function*, artinya mengeneralisasi data yang diterima sebagai masukan agar dapat digunakan sebagai sistem pemetaan berulang untuk

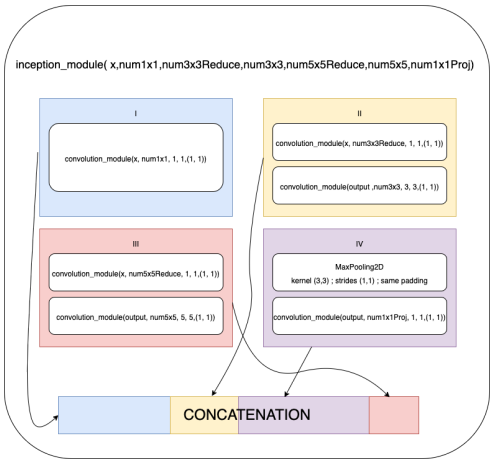

Fig. 4. *Inception module*

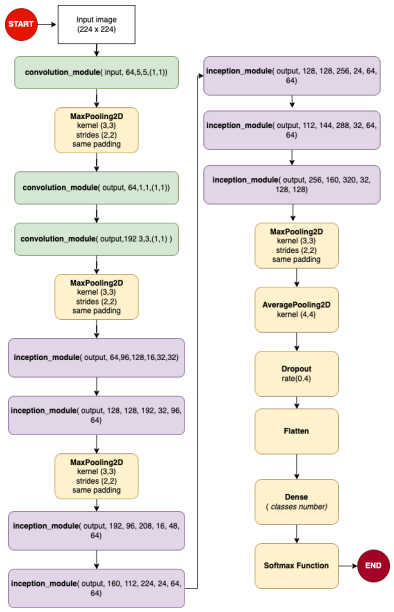

Fig. 5. *Inception architecture*

mengklasifikasi atau memprediksi data baru berikutnya. Sistem ini memiliki banyak tantangan, di antaranya adalah *noise* pada dataset. Pada praktiknya, *deep learning* dioptimasi menggunakan *optimation function*. Fungsi ini berfungsi untuk mencari serangkaian nilai masukan agar menghasilkan nilai keluaran yang merupakan nilai minimum ataupun maksimum dari suatu fungsi objektif. Dalam kasus penelitian ini fungsi objektif yang dimaksud adalah *cost function* [2].

Optimizers adalah salah satu metode untuk meminimalisasi nilai keluaran dari *cost function*. Merupakan fungsi matematika yang bergantung kepada variabel bebas dari model seperti nilai bobot dan *bias*. *Optimizer* mempelajari bagaimana cara untuk mengubah nilai bobot dan *bias* dari *neural network* untuk mengurangi *error* [30]. Pada penelitian ini digunakan 5 jenis *optimizer* yaitu *Stochastic Gradient Descent, Nesterov Accelerated Gradient, Adagrad, Adadelta & Adam*.

$$v_t = \gamma v_{t-1} + \alpha \nabla J(\vartheta - \gamma v_{t-1}) \quad (1)$$

$$\vartheta_{t+1} = \vartheta_t - v_t \quad (2)$$

Keterangan :
v : *velocity*
α : *learning rate*
ϑ : posisi
γ : *hyperparameter*
$\nabla J(\vartheta)$: turunan parsial *cost function*

1) Stochastic Gradient Descent

Gradient adalah ukuran yang menunjukkan pengaruh dari perubahan suatu variabel terhadap nilai keluarannya. Dalam *Deep Learning*, *Gradient* adalah pengaruh perubahan nilai bobot terhadap perubahan nilai *error*. *Gradient Descent* adalah suatu proses *iterative*/berulang yang membawa kepada nilai minimum suatu fungsi [10].

Pada *Stochastic Gradient Descent*, nilai bobot diubah setiap satu data latih selesai dievaluasi. Keuntungan dari metode *SGD* ini adalah prosesnya lebih cepat dibandingkan dengan *Batch Gradient Descent*, perkembangan model juga lebih detail karena perubahan dilakukan setiap satu data selesai dilatih. Kelemahan dari metode ini adalah daya komputasinya tinggi dan banyaknya *noise* yang muncul pada *gradient* karena perubahan yang lebih sering terjadi sehingga menyebabkan tingkat *error* melompat dan tidak berkurang secara bertahap [10].

2) Nesterov Accelerated Gradient

Gradient descent memiliki kelemahan, yaitu model memiliki probabilitas terjebak pada *local minimum*. Oleh karena itu, ditemukanlah metode *momentum* untuk mengatasi hal ini. Namun, penggunaan momentum dapat berdampak buruk terhadap akurasi. Nilai *velocity* yang merupakan akumulasi dari nilai *gradient* bertambah besar setiap iterasinya. Hal ini mencegah model mencapai konvergen. Karena itu dibutuhkan algoritma yang lebih kuat, tidak hanya mempercepat tapi juga memperlambat. Seolah-olah bola yang turun di lembah yang berbukit. Diperlukan algoritma yang mempercepat juga memperlambat bola agar tinggal diam di titik minimum. Lahirlah metode *Nesterov Accelerated Gradient*. Bagaimana *NAG* mengubah bobot dan bias dapat dilihat pada persamaan 1 dan 2.

3) Adagrad

Algoritma *SGD* maupun *NAG* tidak mengubah nilai *learning rate* setiap iterasinya. Perubahan *learning rate* harus diimplementasikan menggunakan metode diluar dari *optimizer* itu sendiri. algoritma *adagrad* mengaplikasikan perubahan *learning rate* dalam metode *optimizer*-nya. Melihat pada persamaan 3, perubahan *learning rate* dilakukan dengan cara membagi

$$\vartheta_{t+1} = \vartheta_t - \sqrt{\frac{\alpha}{G_t + \epsilon}} \times g_t \quad (3)$$

$$G_t = \sum_{i=1}^{t} \frac{\partial L}{\partial W_{t-1}} \quad (4)$$

Keterangan :
g_t : *gradient* pada saat *step* ke t
G_t : akumulasi dari nilai *gradient* sebelumnya.
ϵ : *smoothing term*

nilai tersebut dengan akumulasi nilai *gradient* setiap iterasinya (persamaan 4). Metode ini dimaksudkan agar *learning rate* beradaptasi terhadap jenis fitur. Jenis fitur yang *dense* atau padat dilatih dengan *learning rate* bernilai besar. Setelah proses pelatihan muncul beberapa *dead neuron* atau fitur yang tidak berpengaruh sehingga nilai *feature map* menjadi 0. Fitur yang semula padat menjadi fitur yang *sparse*. Fitur *sparse* cocok bila menggunakan *learning rate* yang bernilai lebih kecil. Fitur pada tahap ini dianggap sebagai fitur-fitur yang cukup berpengaruh terhadap klasifikasi kelas tersebut. Sehingga perubahan nilai bobot dan *bias* dinilai tidak perlu begitu signifikan atau besar.

Salah satu keuntungan dari *adagrad* adalah mengeliminasi keperluan *learning rate tuning*. Pada implementasinya, nilai *learning rate* mula-mula adalah 0.01. Kelemahan dari *adagrad* adalah akumulasi dari nilai *gradient* yang setiap iterasinya bertambah besar, mengakibatkan *learning rate* menjadi semakin kecil sehingga pada akhirnya model tidak mampu mempelajari fitur penting [26].

4) Adadelta

Adadelta adalah salah satu usaha pengembangan dari algoritma *adagrad*, diciptakan dengan tujuan mengatasi *learning rate* yang berkurang secara monoton. *Adagrad* menjumlahkan semua *gradient* pada iterasi sebelumnya, hal ini mengakibatkan nilai *learning rate* berkurang terus menerus. Apabila nilai *learning rate* menjadi 0 pada suatu waktu, model tidak dapat mempelajari pola lain dari data latih.

Ide dasar dari *adadelta* adalah menjumlahkan porsi dari nilai kuadrat *gradient* sebelumnya dengan mengimplementasikan *decay average of square gradient* seperti pada persamaan 5. Dimana *gamma* adalah *decay constant* [27]. *Decay constant* memiliki fungsi yang sama dengan *alpha* pada metode momentum, yaitu mengatur seberapa besar pengaruh akumulasi *gradient* sebelumnya terhadap perubahan *learning rate*. Pada pustaka *keras* nilai tersebut diinisialisasi dengan nilai 0.95. Setelah didapatkan nilai akumulasi *gradient*, nilai akar kuadratnya dicari dengan menggunakan persamaan *6.* Selanjutnya persamaan perubahan dari *adagrad* diubah, yaitu yang sebelumnya menggunakan akumulasi *gradient* dengan menggunakan akar kuadrat seperti pada persamaan 7 dan 9 [27]. Persamaan 8 adalah *RMSprop*, *optimizer* yang dikembangkan bersamaan dengan *Adadelta* dengan tujuan mengatasi kelemahan dari *Adagrad*. Perbedaan dari *RMSprop*

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) g_t^2 \quad (5)$$

$$RMS[g]_t = \sqrt{E[g^2]_t + \epsilon} \quad (6)$$

$$\Delta\vartheta_t = -\frac{RMS[\Delta\vartheta]_{t-1}}{RMS[g]_t} g_t \quad (7)$$

$$\Delta\vartheta_t = -\frac{\alpha}{RMS[g]_t} g_t \quad (8)$$

$$\vartheta_{t+1} = \vartheta_t + \Delta\vartheta_t \quad (9)$$

Keterangan :
E : *running average*
g : *gradient*
γ : Konstanta yang melimit perubahan *gradient*
α : *learning rate*
ϑ : posisi
G : *diagonal matrix*
RMS : *Root Mean Square*

dan *Adadelta* terletak pada nilai pembilangnya, pada *RMSprop* yang digunakan adalah nilai *learning rate*, sementara pada *Adadelta* yang digunakan adalah nilai akar kuadrat dari perubahan posisi iterasi sebelumnya.

5) Adam

Adam adalah singkatan dari *Adaptive Moment Estimation*, merupakan perpaduan antara metode momentum dan *RMSprop*. Pada dasarnya Adam mengambil keuntungan dari metode momentum yaitu *smoothening* dan dari metode *RMSprop* yaitu *learning rate decay*. Persamaan 11 dan 11, menunjukan persamaan *velocity* untuk *smoothening* baik untuk bobot dan juga *bias*. Demikian juga persamaan 12 dan 13 merupakan persamaan *learning rate decay*. *Adam* memadukan kedua persamaan ini untuk mengubah nilai bobot dan *bias* seperti pada persamaan 18 dan 19. Setiap iterasinya *hyperparameter* pada metode *adam* nilainya diubah, mengakibatkan nilai *velocity* dan *learning rate decay* berubah seperti pada persamaan 14, 15, 17, 16 [45].

C. Learning rate scheduler

Pada proses *backpropagation*, *learning rate* berperan untuk menentukan seberapa besar perubahan diaplikasikan terhadap nilai bobot dan *bias*. Penentuan nilai *learning rate* memegang peranan penting dalam proses pembelajaran model. Nilai *learning rate* yang terlalu kecil mengakibatkan model membutuhkan *epoch* yang lebih panjang untuk mencapai konvergen. Sementara penentuan nilai *learning rate* yang terlalu besar mengakibatkan model tidak mengalami konvergen.

Learning rate merupakan bilangan desimal positif dengan rentang nilai diantara 0 hingga 1. Nilai ini mengontrol bagaimana model beradaptasi terhadap fitur masukan. Pada aplikasi program, penggunaan *learning rate* dapat digunakan bersamaan dengan *optimizer* maupun menggunakan *learning*

$$V_{dw} = \beta_1 V_{dw} + (1 - \beta_1) \frac{\partial L}{\partial w} \quad (10)$$

$$V_{db} = \beta_1 V_{db} + (1 - \beta_1) \frac{\partial L}{\partial b} \quad (11)$$

$$S_{dw} = \beta_2 S_{dw} + (1 - \beta_2)(\frac{\partial L}{\partial w})^2 \quad (12)$$

$$S_{db} = \beta_2 S_{dw} + (1 - \beta_2)(\frac{\partial L}{\partial b})^2 \quad (13)$$

$$V'_{dw} = \frac{V_{dw}}{1 - \beta_1^t} \quad (14)$$

$$V'_{db} = \frac{V_{db}}{1 - \beta_1^t} \quad (15)$$

$$S'_{dw} = \frac{S_{dw}}{1 - \beta_2^t} \quad (16)$$

$$S'_{db} = \frac{S_{db}}{1 - \beta_2^t} \quad (17)$$

$$Wt = Wt_{-1} - \frac{\alpha * V_{dw}}{\sqrt{S'_{dw}} + \epsilon} \quad (18)$$

$$bt = bt_{-1} - \frac{\alpha * V_{db}}{\sqrt{S'_{db}} + \epsilon} \quad (19)$$

Keterangan :
$V_{dw,}$: *velocity on respect of w*
V_{dw} : *update of velocity on respect of w*
V_{db} : *velocity on respect of b*
V_{db}' : *update of velocity on respect of b*
β_1 : *learnable parameter 1*
β_1' : *update of learnable parameter 1*
β_2 : *learnable parameter 2*
β_2' : *update of learnable parameter 2*
∂L : *partial derivative* dari *Loss function*
S_{dw} : perubahan posisi (mirip ϑ) terhadap bobot
S_{dw}' : pertambahan dari perubahan posisi (mirip ϑ) terhadap bobot
S_{db} : perubahan posisi (mirip ϑ) terhadap bias
S_{db}' : pertambahan dari perubahan posisi (mirip ϑ) terhadap bias

rate scheduler. *Learning rate tuning* yang dilakukan pada penelitian ini adalah menggunakan teknik *learning rate decay* yaitu teknik *neural network* modern yang memulai pelatihan dengan nilai *learning rate* yang besar lalu secara perlahan nilainya dikurangi hingga mencapai *local minima*.

Pada figure 6 dijelaskan langkah yang diambil model hingga mencapai *local minima* cukup *noisy* dan nampak membutuhkan waktu sedikit lebih lama untuk mencapai konvergen. Sementara pada gambar 7, terlihat pada tahap awal karena dimulai dengan nilai *learning rate* yang cukup besar, maka langkah yang diambil cukup besar. Namun seiring dengan mendekatnya nilai terhadap *local minima*, langkah yang diambil model semakin kecil dan model semakin mendekat kepada nilai *local minima* [22].

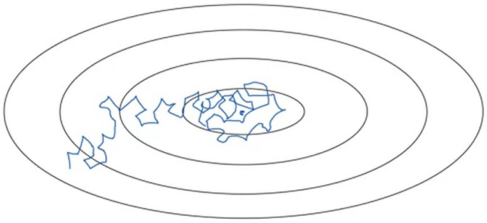

Fig. 6. *Using constant learning rate* [22]

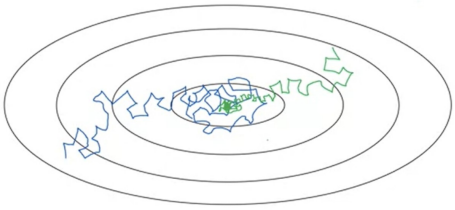

Fig. 7. *Using constant learning decay* [22]

Pada penelitian ini, metode yang digunakan sebagai *learning rate tuning* adalah metode *learning rate decay*. Metode yang digunakan dalam pengujian adalah *step decay, time based decay & exponential weight decay*.

1) Step decay

Mengurangi / mengubah nilai *learning rate* secara konstan setiap interval *epoch* yang telah ditentukan. Persamaan matematika dapat dilihat pada persamaan 20

2) Time based decay

Perubahan nilai *learning rate* berlangsung secara konstan disetiap *epoch*-nya. Persamaan matematika dapat dilihat pada persamaan 21.

3) Exponential weight decay

Metode ini mengaplikasikan fungsi perubahan terhadap model yang bersifat *exponential* setiap *epoch*-nya. Persamaan matematika dapat dilihat pada persamaan 22.

D. Image preprocessing

Image preprocessing adalah langkah-langkah yang diambil untuk memproses gambar sebelum mereka digunakan untuk

$$\alpha_t = \alpha_o * dr^{\frac{1+e}{ed}} \quad (20)$$

Keterangan :
α_t : *learning rate* baru
α_o : *learning rate* sebelumnya
dr : *drop ratio*
e : *epoch number*
ed : *determined interval epoch*

$$\alpha_t = \frac{1}{1 + d * e} * \alpha_o \quad (21)$$

Keterangan :
α_t : *learning rate* baru
α_o : *learning rate* sebelumnya
d : *decay rate*
e : *epoch number*

melatih model.*Image augmentation* adalah manipulasi yang diaplikasikan pada gambar untuk menciptakan versi yang berbeda dari suatu konten yang sama. Tujuannya adalah memperkenalkan model terhadap contoh data yang lebih luas. Perbedaan antara *image preprocessing* dengan *image augmentation* adalah *preprocessing* digunakan pada tahap pelatihan dan tahap pengujian sementara *augmentation* hanya digunakan pada tahap pelatihan. *Image preprocessing* diperlukan untuk membersihkan data gambar yang hendak dipakai saat melatih model, misalnya membuat gambar memiliki dimensi ukuran yang sama.

Pada penelitian ini, metode *preprocessing* yang dipakai adalah *Modified Contrast Limited Adaptive Histogram Equalization*. Pada dasarnya *MCLAHE* adalah *CLAHE* yang diproses lanjut menggunakan metode *gaussian blur*. Proses ini ditujukan untuk mengurangi *noise contrast* yang dihasilkan melalui proses *CLAHE* [1].

1) Image augmentation

Image augmentation adalah teknik untuk mengaplikasikan transformasi pada gambar yang menghasilkan salinan gambar-gambar baru. Dimana gambar-gambar baru tersebut memiliki perbedaan pada aspek tertentu dari gambar asalnya tergantung dari teknik augmentasi yang dipakai. Mengaplikasikan variasi pada gambar tidak mengubah kelas dari objek tersebut, hanya menghasilkan sudut pandang baru terhadap objek. Teknik ini tidak hanya memperbesar ukuran *dataset*, tetapi juga terlibat dalam menambah variasi dari *dataset* yang mengakibatkan model tergeneralisasi lebih baik. Dengan demikian teknik augmentasi mencegah model dari *overfitting*. Terdapat beberapa teknik augmentasi diantaranya adalah *rotation, shifts, flips, brightness* dan *zoom* [40].

2) Lab color channel

Penelitian kali ini menguji pengaruh pemrosesan *MCLAHE* pada dataset terhadap nilai akurasi yang didapat. Metode *CLAHE* adalah metode untuk mendistribusikan kontras pada

$$\alpha_t = \alpha_o * k^{\epsilon} \quad (22)$$

Keterangan :
α_t : *learning rate* baru
α_o : *learning rate* sebelumnya
k : *decay rate*, biasanya dibawah 1, paling umum digunakan adalah 0.95
e : *epoch number*

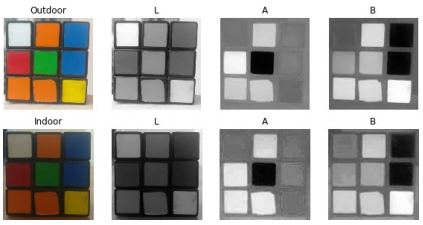

Fig. 8. *Lab color channel example*

gambar. Pendistribusian kontras dilakukan pada channel *L* pada sistem *Lab*, oleh karena itu perlu untuk mengkonversi *RGB* menjadi *Lab*.

Lab color spaces memiliki tiga dimensi, yaitu:

1) *Lightness (L)*
   mengandung intensitas yang terdapat pada gambar. Pada *RGB color spaces* terdapat tiga *channel*, masing-masing *channel* sudah ter-*encoding* data mengenai *brightness* gambar. Pada *LAB - colorspaces* data *brightness* dipisahkan kedalam bentuk *channel* tersendiri yaitu *lightness channel*.
2) *a*
   channel ini mengandung komponen warna dari hijau (*#00FF00*) hingga magenta (*#FF00FF*)
3) *b*
   channel ini mengandung komponen warna dari biru (*#0000FF*) hingga kuning (*#FFFF00*)

3) Modified Contrast Limited Adaptive Histogram Equalization

Contrast Limited Adaptive Histogram Equalization (CLAHE) adalah perkembangan dari metode *AHE*. *AHE* cenderung mengamplifikasi *contrast* pada daerah homogen. *CLAHE* mengatasi masalah ini dengan membatasi amplifikasinya dengan metode *clip limit*. algoritma dari *CLAHE* sendiri mirip dengan metode *AHE* yaitu dengan menggunakan *kernel* yang dilalui ke seluruh *field* gambar. Proses *histogram equalization* berlangsung hanya pada *kernel* tersebut. Namun perbedaannya adalah pada saat perhitungan *PDF*, apabila nilai *PDF* melebihi *clip limit* yang sudah ditentukan, maka nilai lebihnya ditampung dan pada akhir proses dibagi merata ke seluruh *pixel* yang terlibat. Dengan metode ini, histogram sebelum menghitung *cumulative distribution function (CDF)*, melimitasi *contrast* yang berlebih sehingga amplifikasi *noise* diatasi [1] (lihat figure 9).

Pada dasarnya *Modified Contrast Limited Adaptive Histogram Equalization / MCLAHE* adalah *CLAHE* yang dilanjutkan oleh proses *gaussian blurring* dengan *kernel* berukuran 5x5. Hal ini dilakukan untuk menghindari masalah *noise* yang dihasilkan akibat proses *CLAHE* [1].

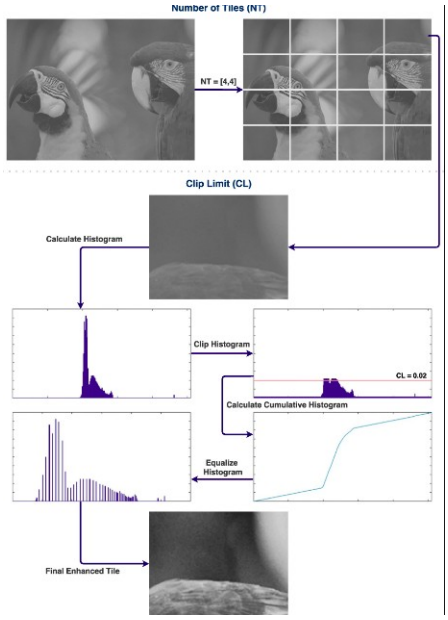

Fig. 9. *CLAHE diagram*

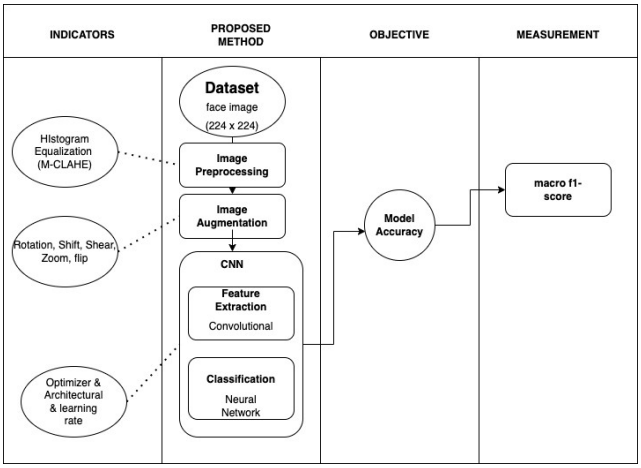

Fig. 10. Kerangka Pemikiran

E. Perancangan Sistem

1) Kerangka Pemikiran

Figur 10 menggambarkan kerangka pemikiran dari metode yang diusulkan untuk melakukan sistem pengenalan wajah *(face recognition)*. Penelitian ini dimulai dengan mempersiapkan *dataset* berupa gambar wajah. Gambar wajah nantinya diproses menggunakan algoritma *MCLAHE*. Sebagai pembanding, dipersiapkan juga *dataset* originalnya, nantinya dataset-dataset ini menjadi data latih untuk arsitektur *CNN* yang dibangun. Selain membandingkan jenis *dataset*, jenis *optimizer*, jenis arsitektur *CNN* dan juga perubahan nilai *learning rate* turut diuji pada penelitian kali ini. Penelitian ini bertujuan untuk melihat hasil akurasi *CNN* dalam melakukan pengenalan wajah. Berikut ini dijelaskan setiap bagian pada gambar 10.

1) *Indicators* adalah variabel yang mempengaruhi hasil dari metode utama. Indikator dinilai pada saat pengujian *CNN* dimulai. Indikator yang diobservasi selama penelitian ini antara lain sebagai berikut.

a) Pemrosesan *MCLAHE* digunakan pada jurnal [1] sebagai pemrosesan awal gambar. Penelitian yang dilakukan pada jurnal [1] menggunakan dataset *Yale* yang merupakan gambar wajah dengan beragam kondisi pencahayaan. *MCLAHE* ditujukan untuk meratakan distribusi intensitas pada gambar. Penelitian ini bertujuan untuk menguji apakah pemrosesan *MCLAHE* juga berpengaruh terhadap hasil akurasi model. *MCLAHE* adalah singkatan dari *modified contrast limited adaptive histogram equalization*. Merupakan metode pemrosesan gambar melalui proses *CLAHE* yang setelah itu diproses kembali dengan *gaussian blur*. *CLAHE* adalah proses *adaptive histogram equalization* yaitu *histogram equalization* yang berlangsung pada *kernel window* yang digeser ke seluruh *field* gambar. Lalu terdapat atribut *clip limit* yang membatasi frekuensi kemunculan suatu *pixel* pada titik tertentu. Lalu kelebihan dari keseluruhan *pixel* tersebut dibagi merata ke seluruh *pixel* pada *kernel* tersebut. Pada penelitian kali ini dilakukan uji coba mengenai tingkat akurasi model dengan data latih yang sudah diproses dengan *MCLAHE* dengan yang belum. Penggunaan metode ini membagi jumlah frekuensi kemunculan *pixel* pada *channel* L pada gambar. Mempengaruhi tingkat *contrast* gambar.

b) *Image augmentation* merupakan metode untuk augmentasi gambar. Proses augmentasi yang diaplikasikan pada penelitian ini adalah transformasi rotasi, *shifting, shearing, zoom & flip*. Tujuan dari transformasi ini adalah untuk memperkenalkan *variant* gambar dari satu kelas yang sama pada model *deep learning*. Hal ini mencegah model untuk mempelajari data latih terlalu detail sehingga mencegah terjadinya *overfitting*. Proses augmentasi gambar mengeneralisasi model yang dihasilkan. Jumlah dataset yang digunakan pada penelitian ini, untuk *Yale* adalah 1647 dan untuk Komnet adalah 810. Jumlah batch yang digunakan saat pelatihan adalah 32. Artinya akan ada 51 iterasi untuk *Yale* dan 25 iterasi untuk Komnet pada 1 *epoch*-nya. Data augmentasi diciptakan secara otomatis dengan menggunakan *ImageGenerator* pada pustaka *keras*. *ImageGenerator* menciptakan data augmentasi setiap iterasi *batch* sejumlah nilai *batch*. Ini artinya data sampel untuk Komnet diciptakan sebanyak 25 x 25 x 100 = 62500 data, sementara data sampel untuk *Yale* diciptakan sebanyak 51 X 51 x 100 = 260100 data. Pengalian dengan 100 dikarenakan nilai *epoch* yang dipakai pada penelitian ini adalah 100.

c) *Architectural* merupakan jenis arsitektur *CNN* yang digunakan saat proses pelatihan model. Jenis arsitektur ini menentukan bagaimana *neural network* mengekstraksi fitur pada gambar yang nantinya dipelajari pada lapisan *artificial neural network* untuk dilakukan proses klasifikasi. Perbedaan pola pada arsitektur *neural network* tentunya hasil fitur yang diperoleh juga berbeda. Penelitian ini bertujuan untuk menganalisa jenis arsitektur yang menghasilkan akurasi paling tinggi.

d) *Optimizers* merupakan *hyperparameter* yang bertujuan untuk menurunkan nilai *loss*. Berkaitan erat dengan *learning rate*, terdapat beberapa metode *optimizer* yang menggunakan sistem *decay* atau mengubah nilai *learning rate* setiap *epoch*-nya. Pada penelitian kali ini, jenis *optimizer* yang dipakai adalah *Stochastic Gradient Descent, Nesterov Accelarated Gradient, Adagrad, Adadelta* dan *Adam*. Tujuan akhir dari *optimizer* adalah menurunkan nilai *loss*. Tujuan dari penelitian ini adalah menguji pengaruh penggunaan *optimizer* terhadap nilai akurasi model pada arsitektur tertentu.

e) *Learning rate* merupakan *hyperparameter* yang menentukan seberapa besar perubahan pada nilai bobot dan *bias* diaplikasikan pada tahap pelatihan. Figur 6 menjelaskan nilai *learning rate* yang konstan mengakibatkan model membutuhkan waktu lebih lama untuk mencapai konvergen, sementara pada figur 7 menjelaskan bahwa perubahan nilai *learning rate* mempercepat model mencapai konvergen. Pada penelitian kali ini pengujian perubahan nilai *learning rate* menggunakan metode *learning rate scheduler* dan diuji pada arsitektur dan *optimizer* tertentu. Persamaan perubahan *learning rate* yang diuji adalah *step decay*, *time based decay* dan *exponential weight decay*. Penelitian ini bertujuan untuk menguji pengaruh perubahan nilai *learning rate* terhadap nilai akurasi dari model yang dilatih pada arsitektur dan *optimizer* tertentu.

2) *Proposed Method* adalah bagian yang menjelaskan proses penelitian dari awal hingga akhir. Proses ini dimulai dengan mempersiapkan *dataset*. *Dataset* yang ada kemudian diproses dengan menggunakan *MCLAHE*, lalu dilanjutkan dengan meng-*upload dataset* original dengan *dataset* yang sudah diproses ke dalam *google drive*. Lalu *dataset* diteruskan kepada model *CNN*. Sebelum data dilatih oleh model, data terlebih dahulu diaugmentasi. Tujuan dari augmentasi adalah mengaplikasikan regularisasi pada model *CNN*. Tujuan dari regularisasi adalah supaya model yang dihasilkan nanti dapat beradaptasi dengan data lain yang beragam jika dibandingkan dengan data latihnya. Model *CNN* kemudian dilatih dan diuji untuk memperoleh hasil akurasi pengenalan wajah.

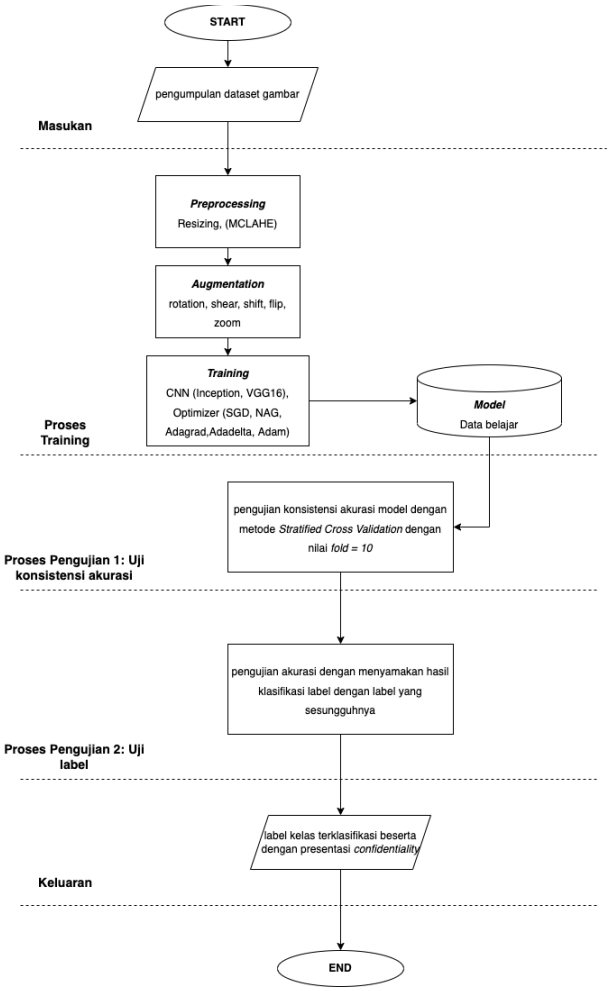

Fig. 11. *global flowchart*

3) *Objectives* adalah bagian yang menjelaskan target yang menjadi acuan pengukuran. Dalam penelitian ini, target tersebut adalah akurasi dari pengenalan wajah oleh *CNN*.

4) *Measurement* adalah satuan ukur yang digunakan untuk mengukur hal-hal yang ada pada bagian *Objectives*. Dalam penelitian ini, karena hendak melakukan *multiclass classification* maka *cost function* yang dipakai adalah *categorical cross entropy*. Pengukuran akurasi menggunakan *macro f1-score*. Pada sistem pengenalan wajah, *false positive* dan *false negative* lebih *crucial* dibandingkan dengan *true positive* dan *true negative*. Karena penting untuk mengetahui mengapa model menebak suatu kelas sebagai kelas yang lain, dibanding hanya mengetahui tebakan benar dan salahnya saja. Oleh karena itu, penelitian ini menggunakan *macro f1-score* sebagai *measurement* [28].

2) Flowchart Global

Penelitian ini menggunakan *CNN* untuk melakukan pengenalan wajah. Setelah dilakukan pelatihan, diharapkan arsitektur yang dibangun dapat menghasilkan akurasi yang baik. Gambar 11 menunjukkan urutan proses global pada penelitian ini dalam bentuk *flowchart*. Tahapan identifikasi wajah terbagi atas tiga proses, yaitu pemrosesan awal, proses pelatihan dan proses pengujian. Pemrosesan awal dilakukan untuk mengaplikasikan algoritma *Modified Contrast Limited Adaptive Histogram Equalization* sesuai dengan yang tertulis pada jurnal [1]. Proses pelatihan dilakukan untuk mendapatkan model *CNN* yang optimal untuk menghasilkan akurasi yang tinggi. Proses pengujian dilakukan untuk menguji generalisasi model. Proses pengujian sendiri terdiri dari 2 tahapan, tahapan pertama adalah pengujian 2 model terbaik dengan metode *cross fold validation* dengan nilai *fold* 10. Sementara pengujian kedua adalah pengujian akurasi terhadap seluruh model.

3) Dataset

Dataset yang digunakan dalam penelitian ini ada dua jenis, yaitu *cropped extended yale face database* dan *komnet dataset*. *Cropped extended yale face database* adalah *extended yale face database* yang diaplikasikan algoritma *Viola-Jones*, sehingga yang dihasilkan adalah gambar wajah yang menghadap ke depan *(frontal face)*. *Cropped extended yale face database* mengandung 38 kelas dengan masing masing kelas terdiri dari 64 gambar dengan beragam jenis iluminasi pencahayaan. Dimensi awal dari *cropped Yale* dataset adalah 168×192 pixel. Dengan format data .PNM *(Portable Any Map Image)*, yang menyimpan satu gambar tanpa kompresi.

Dataset berikutnya yang dipakai pada penelitian ini adalah *komnet dataset*. *Dataset* ini dibuat oleh laboraturium politeknik Bali, mengandung 50 kelas dengan masing masing kelas terdiri dari 24 gambar berukuran 224 x 224 *pixel*. Gambar berupa wajah yang menghadap ke depan *(frontal face)*. Gambar diambil dengan menggunakan media kamera digital, sosial media dan juga kamera *smartphone*. Dimensi awal dari gambar ini adalah 224 × 224 *pixel* dengan format *jpeg*.

III. Hasil dan Pembahasan

A. Hasil Pengujian

Akurasi model tertinggi yang diperoleh dari hasil uji coba dengan arsitektur *Inception* adalah saat pengujian terhadap dataset Komnet dengan *optimizer Adadelta*. Sementara pada arsitektur VGG16 adalah saat pengujian dengan *optimizer NAG* dengan *step decay learning rate scheduler*. Pada pengujian ini, kedua model terbaik tersebut dilakukan uji *stratified cross validation* dengan nilai *fold* sebesar 10. Lalu pengujian akurasi model dilakukan disetiap iterasinya. Figur 12 menggambarkan fluktuasi nilai akurasi dari arsitektur VGG16 sementara figur 13 menggambarkan flukutasi nilai akurasi arsitektur *Inception*. Dapat disimpulkan dari kedua figur tersebut

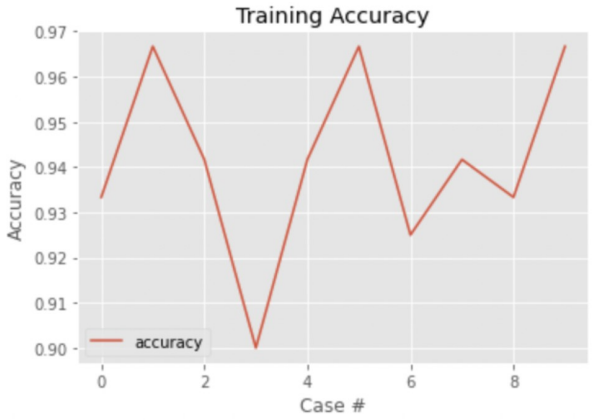

Fig. 12. uji konsistensi *VGG16*

Fig. 13. uji konsistensi *Inception*

bahwa *Inception* memiliki tingkat akurasi dan *robustness* lebih baik dibandingkan dengan VGG16.

IV. Kesimpulan

1) Penggunaan pemrosesan *MCLAHE* memberikan hasil akurasi validasi yang lebih tidak fluktuatif. Penggunaan *MCLAHE* disimpulkan mengurangi resiko *overfitting*. Namun penggunaan *MCLAHE* secara umum menurunkan hasil akurasi.
2) Penggunaan *optimizer* sangat berpengaruh pada pencapaian akurasi model. Pencapaian akurasi tertinggi dicapai oleh model *Inception* dengan *optimizer adadelta* dengan akurasi pelatihan 100%
3) Penggunaan algoritma perubahan nilai learning rate mempengaruhi tingkat akurasi model. Pada pengujian terdapat 28 dari 48 total kasus uji (58.3%) yang mengalami peningkatan nilai akurasi akibat penggunaan nilai learning rate yang dinamis. Berikut pasangan *optimizer* beserta *learning rate scheduler* yang meningkatkan nilai akurasi mode.
    a) *Adam* : *step decay, learning rate scheduler*
    b) *SGD,NAG, Adagrad* : *Time based learning rate scheduler*

4) Arsitektur *Inception* menghasilkan akurasi yang lebih baik dibandingkan dengan VGG16. Hal ini diperkirakan karena kriteria *Inception* yang menggunakan jenis filter yang beragam, serta mekanisme arsitektur yang bekerja secara paralel, sehingga memungkinkan model latih mempelajari fitur detail yang mungkin tidak diperoleh pada pelatihan dengan arsitektur VGG16, arena VGG16 hanya menggunakan 1 jenis filter.

REFERENCES

[1] Bendjilali Ridha Ilyas, Beladgham Mohammed, Merit Khaled and Taleb-Ahmed Abdelmalik, *"Illumination-robust face recognition based on deep convolutional neural networks architecture"*, *Indonesian Journal of Electrical Engineering and Computer Science*, May 2020.

[2] Brownlee, Jason PhD. *Why Optimization Is Important in Machine Learning*,https://machinelearningmastery.com/why-optimization-is-important-in-machine-learning/ (accessed February 12, 2022)

[3] Yaqub Muhammad, Feng Jinchao, Zia Sultan, Arshid Kaleem, Jia Kebin, Rehman Zaka Ur adn Mehmood Atif *"State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images"*, *Brain Sciences*, published July 3, 2020.

[4] Samar S. Mohammed, Wael A. Mohamed, A.T. Khalil and A.S. Mora, *"Deep Learning Face Detection and Recognition"*, *Iternational journal of electronics and telecommunications*, June 2019.

[5] KB. Pranav, J. Manikandan, *"Design and Evaluation of a Real-Time Face Recognition System using Convolutional Neural Networks"*, *Article in Procedia Computer Science 171 (2020) 1651–1659*,2020.

[6] Coskun Musab,Ucar Aysegul, Yildirm Ozal and Demir Yakup, *"Face Recognition Based on Convolutional Neural Network"*, *Conference Paper*, November, 2021.

[7] Zhiming Xie, Junjie Li and Hui Shi, *"A Face Recognition Method Based on CNN"*, *2019 High Performance Computing and Computational Intelligence Conference*,2019

[8] C. Beumier, "Face Recognition," *Natl. Sci. Technol. Counc.*, pp. 92–99, 2001.

[9] Bhardwaj, Anjali. *What is a Perceptron? – Basics of Neural Networks*. https://towardsdatascience.com/what-is-a-perceptron-basics-of-neural-networks-c4cfea20c590 (accessed September 13,2021 12:30:34)

[10] Donges, Niklas. *Gradient Descent: An Introduction to 1 of Machine Learning's Most Popular Algorithms*. https://builtin.com/data-science/gradient-descent (accessed September 14, 2021 11:40:23)

[11] Pandey, Paul. *Understanding the Mathematics behind Gradient Descent*. https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e (accessed September 14, 2021 11:46:30)

[12] Shah, Saily. *Cost Function is No Rocket Science!* https://www.analyticsvidhya.com/blog/2021/02/cost-function-is-no-rocket-science/ (accessed September 14, 2021 12:45:12)

[13] B.Dickson. *"What are convolutional neural networks* (CNN)?".bdtechtalk.com. https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/ (accessed May 4, 2021 14:55:23)

[14] IBM Cloud Education. *Convolutional Neural Networks* https://www.ibm.com/cloud/learn/convolutional-neural-networks (accessed October 25, 2021 13:30:30)

[15] IBM Cloud Education. *Neural Networks*. https://www.ibm.com/cloud/learn/neural-networks (accessed September 13, 2021 17:17:23)
bib

[16] Jeong, Jiwon. *The Most Intuitive and Easiest Guide for Convolutional Neural Network* https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480 (accessed October 25, 2021 13:36 54)

[17] Saxena, Shipra. *Introduction to Batch Normalization* https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/ (accessed October 25, 2021 17:43:42)

[18] Mustafa. *Optimizers in Deep Learning* https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0 (accessed October 31, 2021 19:49:50)

[19] Simonyan. Karen and Zisserman. Andrew, *"VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE SCALE IMAGE RECOGNITION" conference paper at ICLR 2015, April 10 2015*

[20] Sharma, Pulkit. *"A Comprehensive Tutorial to learn Convolutional Neural Networks from Scratch (deeplearning.ai Course #4)"*. https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/ (accessed October 25, 2021 23:54:50)

[21] Leung, Kenneth. *"Micro, Macro Weighted Averages of F1 Score, Clearly Explained"*. https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f (accessed March 3, 2022 13:31:00)

[22] Haswani, Vaibhav. *"Learning Rate Decay and methods in Deep Learning"*.https://medium.com/analytics-vidhya/learning-rate-decay-and-methods-in-deep-learning-2cee564f910b: :text=Learning rate decay is a,help 0both optimization and generalization (accessed March 3, 2022 13:17:00)

[23] Kumar, Ahlad. *"Lecture: CNN Architectures (AlexNet, VGGNet, Inception ResNet)"*. https://www.youtube.com/watch?v=CNNnzl8HIIU (accessed October 26, 2021 13:24:20)

[24] Rosebrock, Adrian.2017.*Deep Learning for Computer Vision with Python*. PyImageSearch

[25] Rosebrock, Adrian.2017.*Deep Learning for Computer Vision with Python*. pp. 20 - 22.

[26] Ruder, Sebastian.2016.*An overview of gradient descent optimization algorithms*.https://www.notion.so/Gradient-with-momentum-b1b93b72ff71402ebd6c47b49a408f81 (accessed November 1, 2021 (16:28:00)

[27] Kumawat, Neha. *Continuing on Adaptive Method: ADADELTA and RMSProp*. https://insideaiml.com/blog/Adagrad-and-Adadelta-optimizer:-In-depth-explanation-1052 (accessed January 9,2022 (21:42:00))

[28] Huilgol, Purva. *Accuracy vs. F1-Score*. https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2: :text=Accuracy is used when the,and False Positives are crucialtext=In most real life classification,to evaluate our model on (accessed February 12 2022, 22:39:00)

[29] Ilyas, Ridwan. *Penjelasan Algoritma Backpropagation Mengupdate Bobot — Jurnal Kelas*. https://www.youtube.com/watch?v=OeowRowEbBc (accessed November 7, 2021 11:41:00)

[30] Gupta, Dishashree. *Fundamentals of Deep Learning – Activation Functions and When to Use Them?*.https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/ (accessed November 7, 2021 11:45:00)

[31] Naik, Krish. *Deep Learning-All Optimizers In One Video-SGD with Momentum,Adagrad,Adadelta,RMSprop,Adam Optimizers*. https://www.youtube.com/watch?v=TudQZtgpoHk (accessed November 7, 2021 13:08:00)

[32] opencv. *Color conversions*. https://docs.opencv.org/3.4.15/de/d25/ imgproc_color_conversions.html (accessed November 7, 2021 15:26:00)

[33] I. Goodfellow, Y. Bengio, and A. Courville,*Deep Learning, An MIT Press book*,2016. [Online]. Available : http://www.deeplearningbook.org (accessed November 7,2021 16:57:00)

[34] S. Khan, H. Rahmani, Syed Afaq Ali Shah, M. Bennamoun, *"A Guide to Convolutional Neural Networks for Computer Vision"*, 1*5ted, GérardMedioniandSvenDickinson, Ed.California* : *MorganandClaypool,* 2018*, pp.*45*,* 53*,* 56*,* 67 80.

[35] Kapoor, Namrata. *Weight Initialization Techniques-What best works for you*.https://www.numpyninja.com/post/weight-initialization-techniques (accessed November 7,2021 20:15:00)

[36] Taunk, Dhaval. *L1 vs L2 Regularization: The intuitive difference*.https://medium.com/analytics-vidhya/l1-vs-l2-regularization-which-is-better-d01068e6658c (accessed November 7, 2021 21:40:00)

[37] H. Habibi Aghdam and E. Jahani Heravi, *Guide to Convolutional Neural Networks*, 1 st ed. Switzerland: Springer International Publishing AG, 2017, pp. 108-111, 118-120.

[38] Lina, Qobylatul. *Implementasi Deep Learning Menggunakan Convolutional Neural Network untuk Klasifikasi Gambar (Mata Juling dan Mata Normal) dengan R*. https://medium.com/@16611110/implementasi-deep-learning-menggunakan-convolutional-neural-network-untuk-klasifikasi-gambar-mata-87dcc0ad26e0 (accessed November 16, 2021 22:39:00)

[39] H. Kinsley and D. Kukieła, *"Neural Networks from Scratch in Python"*, 1 st ed. Harrison Kinsley, 2020, pp. 108, 333-358.

[40] Bhandari Aniruddha, *Image Augmentation on the fly using Keras Image-DataGenerator*. https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/ (accessed November 24, 2021 18:23:00)

[41] Sudhakar, Shreenidhi. *Histogram Equalization*. https://towardsdatascience.com/histogram-equalization-5d1013626e64 (accessed November 25,2021 12:16:00)

[42] Frickle, Tobin. *Rotation by Shearing*. https://www.ocf.berkeley.edu/ fricke/projects/ israel/ paeth/rotation$_{b}$y$_{s}$*hearing.html*(*accessedNovember*27, 202112 : 15 : 00)

[43] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, 3 rd ed. New Jersey: Pearson Prentice Hall, 2008, pp.91, pp.122-126

[44] H. Habibi Aghdam and E. Jahani Heravi, *Guide to Convolutional Neural Networks*, 1 st ed. Switzerland: Springer International Publishing AG, 2017, pp. 108-111, 118-120.

[45] Naik Krish, *Tutorial 14 - Stochastic Gradient Descent with Momentum*, https://www.youtube.com/watch?v=CKLwvuKWQjo (accessed February 7. 2022 23:41:00)

[46] Prof Balasubramainan Vineeth, *Explaining CNNs: Class Attribution Map Methods* , https://www.youtube.com/watch?v=VmbBnSv3otc (accessed February 7. 2-22 22:00:00)