JEPIN
(Jurnal Edukasi dan Penelitian Informatika)
ISSN(e): 2548-9364 / ISSN(p) : 2460-0741

Vol. 9
No. 2
Agustus
2023

Analisis Pemilihan *Optimizer* dalam Arsitektur *Convolution Neural Network* VGG16 dan Inception untuk Sistem Pengenalan Wajah

Ken Ratri Retno Wardani[#1], Hanjaya Suryalim[#2], Ventje J. Lewi Engel [#3], Hans Christian[#4]
[#]*Program Studi Informatika, Institut Teknologi Harapan Bangsa*
Jalan Dipati Ukur 80-84, Bandung, Indonesia
[1]ken_ratri@ithb.ac.id
[2]hanjayasuryalim@gmail.com
[3]ventje@ithb.ac.id
[4]hans_christian@ithb.ac.id

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 arsitektur *Convolution Neural Network* (CNN), tersusun dari lapisan-lapisan konvolusi yang diikuti oleh *fully connected layer*. Lapisan konvolusi ini yang bertanggungjawab atas proses ekstraksi fitur pada citra yang akan digunakan untuk klasifikasi citra tersebut. Pada penelitian ini diuji dua jenis arsitektur CNN yaitu VGG16 dan Inception untuk mengukur akurasi pengenalan wajah. 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 akurasi. *Optimizer* bekerja dengan cara mengubah nilai bobot dan bias saat proses *backpropagation* dengan tujuan menghasilkan nilai *error* yang minimum. Setiap *optimizer* memiliki algoritma yang unik. Pengujian menggunakan 2 dataset yaitu Komnet dan Yale, serta melakukan pengujian pengaruh *preprocessing* MCLAHE terhadap akurasi. Hasil akurasi tertinggi yang dicapai adalah arsitektur Inception dengan *optimizer* AdaDelta pada dataset Komnet+MCLAHE. Akurasi pada tahap pelatihan mencapai 98%. Rata-rata akurasi setelah model diuji dengan 10-*fold cross validation* adalah 99.3%.

Kata kunci — *face recognition*, CNN, MCLAHE, *optimizer*, AdaDelta, Inception, VGG16

I. PENDAHULUAN

Deteksi wajah adalah teknik komputasi berbasis pembelajaran mesin untuk mendeteksi keberadaan wajah dalam suatu citra. Salah satu aplikasi deteksi wajah adalah pengenalan wajah. Dalam sistem pengenalan wajah, tidak hanya mendeteksi area wajah tetapi dapat menentukan wajah siapa yang ada dalam citra. Sistem ini termasuk kategori klasifikasi citra yaitu memprediksi kelas suatu objek dalam suatu citra. Klasifikasi 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 jika dibandingkan metode *machine learning* konvensional [1].

Penggunaan CNN dalam kasus uji terhadap dataset Yale dengan arsitektur Inception mampu memberikan akurasi sebesar 99.89%. Dataset Yale memuat data citra *frontal face* dengan kondisi pencahayaan yang beragam oleh sebab dilakukan proses *Modified Contrast Limited Adaptive Histogram Equalization* (MCLAHE) untuk meningkatkan kontras global dari citra yang memiliki rentang nilai intensitas yang sempit. Penerapan MCLAHE memberikan hasil yang lebih optimal [1].

Saat proses pelatihan, model *deep learning* perlu memodifikasi atribut seperti bobot dan *learning rate* untuk membantu meminimalkan *loss function* dan meningkatkan akurasi. *Optimizer* digunakan untuk meningkatkan kinerja dari *deep learning* [2]. Masalah pemilihan bobot yang tepat untuk model adalah tugas yang rumit karena model *deep learning* umumnya terdiri dari jutaan parameter.

Kinerja dari CNN tergantung pada beberapa parameter yaitu inisialisasi bobot, *optimizer*, *batch size*, epoh, *learning rate*, fungsi aktivasi, *loss function*, arsitektur jaringan, kualitas data masukan, dan kombinasinya [3]. Di dalam persoalan segmentasi ataupun klasifikasi objek, pengujian dengan menggunakan satu *optimizer* tergolong pengujian yang bersifat lemah kecuali jika memiliki argumentasi yang cukup kuat untuk menggunakan algoritma *optimizer* tertentu. Oleh sebab itu pemilihan *optimizer* merupakan proses yang penting.

Dalam satu penelitian dilakukan pengujian 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*

Submitted 16-12-2022; Revised 21-07-2023; Accepted 23-07-2023

*Momentum (*Nadam*)*, dan Nesterov *Accelerated Gradient (*NAG*)*. Diperoleh hasil terbaik adalah Adam dengan akurasi 99.2% [3]. Hal ini menyebabkan kebutuhan untuk memilih algoritma *optimizer* yang tepat sesuai dengan arsitektur dan dataset yang digunakan.

Pengembangkan arsitektur CNN 15 lapisan dengan menggunakan *optimizer* SGD dan dataset Faces96 mencapai akurasi 99.67 % [4]. Penelitian lain melakukan penyeteman *hyperparameter* pada arsitektur *CNN* menghasilkan akurasi terbaik 98.75% [5]. Pengaruh penambahan normalisasi *batch* terhadap akurasi membantu memitigasi dampak dari *internal covariate shift*. Hal ini terjadi akibat perubahan dalam distribusi input maka lapisan *hidden* mencoba untuk belajar beradaptasi dengan distribusi baru sehingga memperlambat proses pelatihan untuk mencapai konvergen ke minimum global. Akurasi tertinggi yang dicapai adalah 94.8 % [6]. Pengaruh perubahan jumlah neuron di lapisan konvolusi pada arsitektur LeNet-5 memberikan akurasi tertinggi dengan jumlah neuron masing-masing 36, 76, dan 1024 [7].

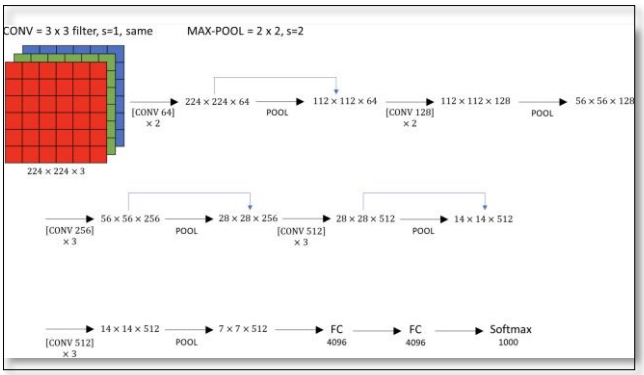
Dalam penelitian ini bertujuan membuat sistem pengenalan wajah dengan menguji 2 model yaitu Inception dan VGG16 dengan 5 jenis optimizer yaitu SGD, Nadam, Adagrad, Adadelta, dan Adam. Dataset yang digunakan Komnet dan Yale. Selain itu akan menguji *preprocessing* MCLAHE terhadap akurasi model.

II. TINJAUAN PUSTAKA

A. Convolutional Neural Network

Convolutional Neural Network (CNN) atau yang biasa disebut juga ConvNets diperkenalkan pada tahun 1980 oleh Yann Lecun. CNN adalah pengembangan dari *multilayer perceptron* berupa *neural network*, khusus untuk memproses data yang memiliki topologi seperti *grid*. Contoh data yang diproses yaitu data *time-domain* sebagai *grid* satu dimensi dimana pengambilan sampel pada interval waktu dan data citra sebagai *grid* piksel dua dimensi.

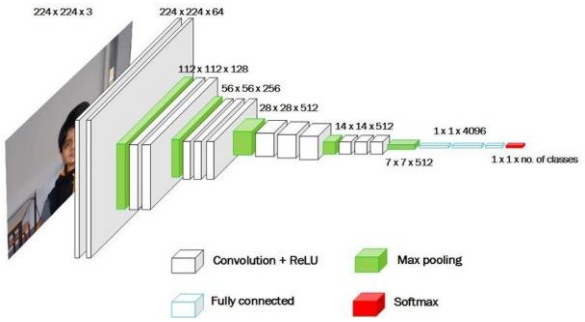
CNN memiliki struktur yang hampir mirip dengan *Feed Forward Neural Network* (FFNN) oleh karena itu CNN memiliki lapisan konvolusional dan beberapa lapisan lain [8]. CNN memiliki tiga jenis lapisan yang membentuk arsitektur yaitu: *convolutional*, *pooling*, dan *dense/fully connected*. Lapisan konvolusi adalah bagian inti dari CNN, di sini sebagian besar proses komputasi terjadi. Pada tahapan ini, masukan adalah citra berwarna 3 *channel*, dalam bentuk matriks piksel 3 dimensi yang merepresentasikan tinggi, lebar, dan kedalaman citra RGB. Kemudian dilakukan proses konvolusi dengan melibatkan pendeteksi fitur, dikenal sebagai kernel atau *filter*. Keluaran hasil konvolusi disebut sebagai *feature map* [9]. Ukuran *filter* dapat ditentukan oleh peneliti tergantung kepada fitur yang ingin dipelajari dari citra masukan. *Filter* belajar memperbaharui nilainya pada tahap pelatihan propagasi *backward*.



Gambar. 1 Struktur VGG16 [8]

1) VGG16: VGG16 memiliki nama tersebut karena memiliki 16 lapisan yaitu 13 lapisan konvolusi dan 3 lapisan *fully connected*, lihat Gambar 1. Hal unik dari VGG16 berfokus pada lapisan konvolusi dengan dimensi filter 3 x 3 dengan *stride* 1, dan *padding same* untuk menjaga dimensi citra. Pada lapisan pooling dilakukan subsampling dengan faktor 2 dan *stride* sebesar 2. *Pengaturan* konvolusi dan *Max pooling* 2x2 konsisten untuk seluruh arsitektur [10][11].

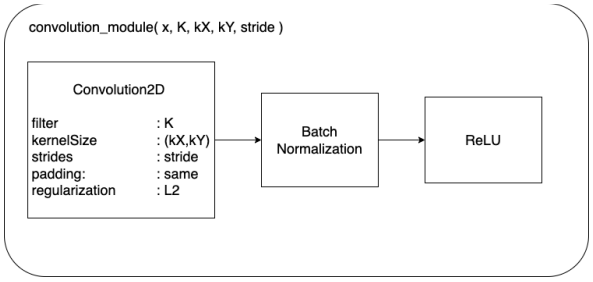
Dimensi data citra masukan pada arsitektur VGG16 adalah 224 × 224 × 3 *channel*, lihat Gambar 2. Pada bagian akhir memiliki lapisan *fully connected* dan diikuti fungsi aktivasi Softmax untuk luaran. Pada penelitian ini, akhir setiap blok konvolusi, ditambahkan lapisan *dropout* dan *batch normalization* sebagai metode regularisasi [12].



Gambar. 2 Lapisan VGG16 [8]

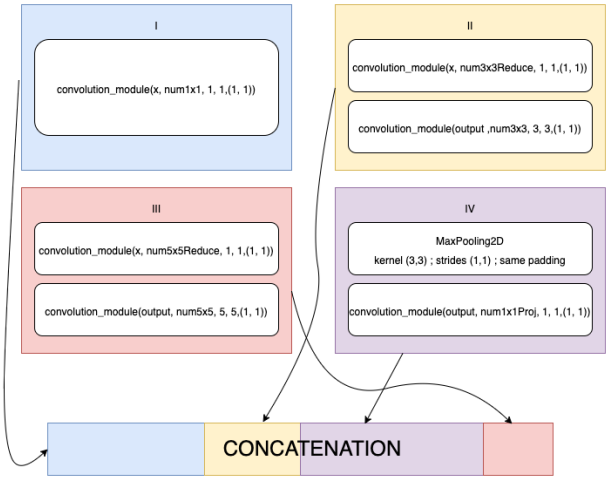
2) *Inception:* Dikenal juga dengan nama "GoogLeNet", ide utama dari arsitektur Inception adalah penggunaan lebih dari satu dimensi filter dalam satu kali konvolusi yaitu 1x1, 3x3, dan 5x5 (konvolusi multi-skala) dan juga penggunaan *max-pooling* 3x3 dengan *stride* 1. Ketiga filter ini berjalan secara paralel dan *feature map* hasil dari setiap filter konvolusi digabungkan menjadi satu kesatuan [10].

Pada arsitektur Inception terdapat dua jenis modul (kumpulan beberapa lapisan yang dipakai berulang), yaitu modul konvolusi dan Inception, lihat Gambar 3. Modul Konvolusi adalah fungsi dengan parameter jumlah *filter*, ukuran *filter*, nilai *stride*, *padding*, dan regularisasi L2.



Gambar. 3 *Modul Konvolusi*

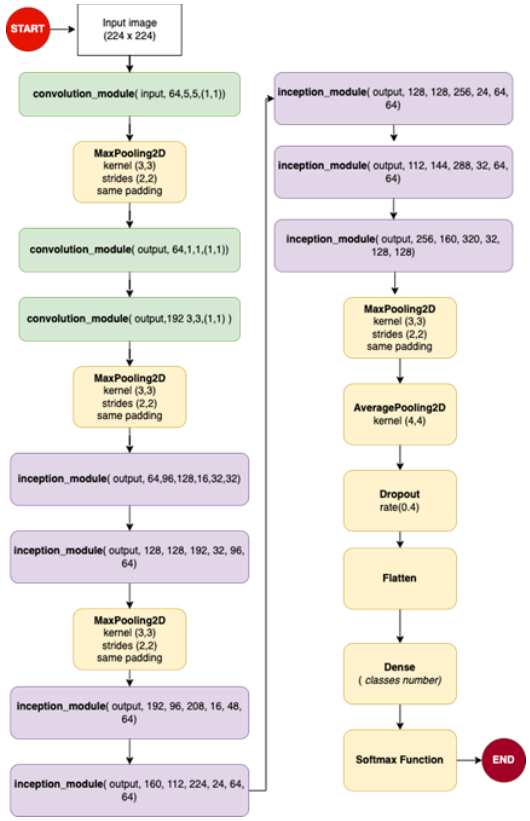
Dalam arsitektur Inception terdapat 7 modul Inception dan 3 modul konvolusi. Proses penggabungan *feature map* berlangsung pada modul Inception. Pada Gambar 4, konvolusi berdimensi 1×1, disebut juga "*Network In Network*". Tujuan dari konvolusi ini adalah mengurangi dimensi data yang melewati jaringan. Meskipun *filter* 1x1 tidak mempelajari pola spasial apa pun di dalam citra, tetapi *filter* ini mempelajari pola di seluruh kedalaman citra. Konvolusi 3x3 dan 5x5 bertujuan untuk mempelajari pola spasial dari citra masukan pada semua dimensi (tinggi, lebar, dan kedalaman). Pemanfaatan berbagai ukuran *filter* menyebabkan model memiliki kemampuan mengekstraksi fitur dari data *input* pada berbagai skala. Gambar 5 struktur arsitektur Inception secara keseluruhan.



Gambar. 4 Modul Inception

A. *Optimizer*

Optimasi berfungsi untuk mencari serangkaian nilai masukan agar menghasilkan nilai keluaran yang merupakan nilai minimum ataupun maksimum dari suatu fungsi. Dalam kasus penelitian ini fungsi yang dimaksud adalah *cost function* [2]. *Optimizer* adalah algoritma yang digunakan untuk meminimalisasi *loss function*. *Optimizer* mempelajari bagaimana menyesuaikan parameter model agar model dapat memberikan prediksi yang lebih baik dan sesuai dengan data yang digunakan selama pelatihan.



Gambar. 5 Arsitektur Inception

1) Stochastic Gradient Descent: Algoritma Stochastic Gradient Descent (SGD) digunakan untuk mencari nilai bobot optimal yang dapat meminimalkan fungsi biaya atau memaksimalkan fungsi tujuan dari model. SGD hanya memerlukan satu data pelatihan pada setiap iterasi sehingga keuntungan menghemat memori tetapi karena proses stokastik (acak) menyebabkan banyaknya *noise* yang muncul pada gradient, fluktuasi ini membuat konvergensi yang lebih lambat [13].

$$Vt = \gamma V\_(t-1) + \alpha \nabla J\_((v - \gamma V\_(t-1)))) \quad (1)$$

$$V\_(t+1) = V\_(t-1) + V\_t \quad (2)$$

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

2) Nesterov Accelerated Gradient: SGD memiliki kelemahan, yaitu fluktuasi stokastik, sehingga biasanya diperlukan lebih banyak iterasi atau lebih banyak penyeteman parameter untuk mendapatkan konvergensi yang baik. Algoritma *Nesterov Accelerated Gradient* (NAG) dikembangkan untuk mengatasi hal itu. Momentum dapat menyebabkan penyimpangan yang berlebihan terutama saat pendekatan ke titik optimum. NAG mengatasi masalah ini dengan memodifikasi pembaruan momentum, memperbarui posisi dengan mempertimbangkan momen yang dihitung sebelumnya.

NAG memperkirakan posisi langkah mendatang menggunakan pembaruan momentum, lalu menghitung gradien pada estimasi tersebut, lihat persamaan 1 dan persamaan 2.

3) Adaptive Gradient: Algoritma SGD maupun NAG tidak mengubah nilai *learning rate* setiap iterasinya. Perubahan learning rate harus menggunakan metode diluar dari optimizer tersebut. Algoritma *Adaptive Gradient* (AdaGrad) mengaplikasikan perubahan learning rate di dalam *optimizer*. Idea dasar adalah memberikan bobot yang berbeda untuk setiap parameter berdasarkan frekuensi gradien masing-masing parameter selama proses pelatihan. Parameter yang jarang muncul atau memiliki gradien yang jarang diperbarui akan memiliki *learning rate* yang lebih tinggi. Sebaliknya, parameter yang sering muncul atau memiliki gradien yang sering diperbarui akan memiliki *learning rate* yang lebih rendah untuk mengurangi fluktuasi.

Pada persamaan 3, perubahan *learning rate* dilakukan dengan cara membagi nilai tersebut dengan akumulasi nilai gradient setiap iterasinya (persamaan 4). Metode ini dimaksudkan agar *learning rate* beradaptasi terhadap jenis fitur.

Salah satu keuntungan AdaGrad adalah tidak perlu menyetel kecepatan pembelajaran secara manual. Kelemahan dari adagrad adalah akumulasi dari nilai gradient pada setiap iterasi bertambah besar, mengakibatkan *learning rate* menjadi semakin kecil sehingga pada akhirnya model tidak mampu mempelajari fitur penting [14].

$$V\_(t+1) = V\_t - \sqrt{(\alpha/(G\_t+\epsilon))}.\, g\_t \quad (3)$$
$$G\_t = \sum(i=1)\partial L/(\partial W\_(t-1)) \quad (4)$$

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

4) AdaDelta: *AdaDelta* adalah algoritma optimisasi adaptif yang merupakan pengembangan dari *AdaGrad*, *learning rate* untuk setiap parameter disesuaikan secara individu. *AdaDelta* dirancang untuk mengatasi masalah learning rate yang mengecil secara berlebihan dan menjadi sangat kecil selama pelatihan. Apabila nilai *learning rate* menjadi 0 pada suatu waktu, model tidak dapat mempelajari pola lain dari data latih. Proses utama *AdaDelta* adalah menggunakan akumulasi kuadrat gradien yang lebih "terbatas" dari *AdaGrad* untuk mengurangi masalah *learning rate* yang mengecil terlalu cepat. Ide dasarnya adalah membatasi jendela akumulasi gradien sebelumnya ke dalam beberapa ukuran tetap. Jumlah gradien didefinisikan secara rekursif sebagai rata-rata yang menurun dari semua gradien kuadrat sebelumnya. Untuk menghitung gradien saat ini dengan persamaan 5, dimana gamma adalah *decay constant* [15], 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.

Perubahan dari Adagrad sebelumnya menggunakan akumulasi gradient persamaan 6, 7,8 dan 9 [15].

$$E[g\^2\ ]\_t = \gamma E[g\^2\ ]\_(t-1) + (1-\gamma) gt\^2 \quad (5)$$
$$RMS[g]\_t = \sqrt{(E[g\^2\ ]\_t + \epsilon)} \quad (6)$$
$$\Delta\theta\_t = -(RMS[\Delta\theta]\_(t-1))/(RMS[g]\_t\ )\ g\_t \quad (7)$$
$$\Delta\theta\_t = -\alpha/(RMS[g]\_t\ )\ g\_t \quad (8)$$
$$\theta\_(t+1) = \theta\_t + \Delta\theta\_t \quad (9)$$

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

5) Adam: *Adaptive Moment Estimation* (Adam) menggabungkan dua teknik yaitu Momentum smoothening dan *Adagrad* untuk mencapai konvergensi yang cepat dan stabil saat pelatihan. Adam dapat memberikan percepatan dalam mengoptimalkan model dan mengatasi masalah seperti *learning rate* yang mengecil terlalu cepat atau fluktuasi *learning rate* yang besar. Adam memerlukan *hyperparameter* tambahan seperti β1, β2, dan *learning rate*. Persamaan 10 dan 11 menunjukkan perhitungan momen pertama untuk bobot dan bias. Persamaan 12 dan 13 menunjukkan perhitungan momen kedua. Adam memadukan kedua persamaan ini untuk mengoreksi nilai bobot dan bias. Setiap iterasi nilai *hyperparameter* diubah, hal ini mengakibatkan nilai *velocity* dan *learning rate decay* berubah, lihat persamaan 14, 15, 16, dan 17 [16]. Perbaharui parameter persamaan 18 dan 19.

$$V\_dw = \beta\_1\, V\_dw + (1-\beta\_1)\ (\partial L)/\partial w \quad (10)$$
$$V\_db = \beta\_1\, V\_db + (1-\beta\_1)\ (\partial L)/\partial b \quad (11)$$
$$S\_dw = \beta\_2\, S\_dw + (1-\beta\_2)\ (\partial L/\partial w)\^2 \quad (12)$$
$$S\_db = \beta\_2\, S\_db + (1-\beta\_2)\ (\partial L/\partial b)\^2 \quad (13)$$
$$V\_dw\^\prime = V\_dw/(1-\beta\_1\^t) \quad (14)$$
$$V\_db\^\prime = V\_db/(1-\beta\_1\^t) \quad (15)$$
$$S\_dw\^\prime = S\_dw/(1-\beta\_2\^t) \quad (16)$$
$$S\_db\^\prime = S\_db/(1-\beta\_2\^t) \quad (17)$$
$$W\_t = W\_(t-1) - (\alpha * V\_dw)/(\sqrt{(S\_dw')}+\epsilon) \quad (18)$$
$$b\_t = b\_(t-1) - (\alpha * V\_db)/(\sqrt{(S\_db')}+\epsilon) \quad (19)$$

V_dw : *velocity on respect of* w
V_dw′ : *update velocity on respect of w*
V_db : *velocity on respect of b*
V_db′ : *update velocity on respect of b*
B1 : *learnable parameter* 1
B1′ : *update of learnable parameter* 1
β2 : *learnable parameter* 2
β2′ : *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

B. Learning Rate Scheduler

Penentuan nilai *learning rate* memegang peranan penting dalam proses pembelajaran model. Nilai *learning rate* yang terlalu kecil mengakibatkan model membutuhkan epoh yang lebih panjang untuk mencapai konvergen. Sementara penentuan nilai *learning rate* yang terlalu besar mengakibatkan model tidak mengalami konvergen.

Learning rate scheduler adalah teknik dalam optimisasi yang digunakan untuk mengatur laju pembelajaran (*learning* rate) secara dinamis selama proses pelatihan. Tujuan dari *learning rate scheduler* adalah untuk meningkatkan efisiensi pelatihan model dan mencapai konvergensi yang lebih cepat dengan mengurangi fluktuasi atau masalah penyimpangan (*overshooting)* yang terkait dengan *learning rate* yang statis.

Learning rate tuning yang dilakukan pada penelitian ini adalah menggunakan teknik *learning rate decay* yaitu memulai pelatihan dengan nilai *learning rate* yang besar lalu secara perlahan nilainya dikurangi hingga mencapai *local minima*. Metode yang digunakan dalam pengujian adalah *Step decay, Time-based decay,* dan *Exponential weight decay*.

1) Step decay: Mengurangi laju *learning rate* secara diskrit setelah mencapai jumlah epoh tertentu, lihat persamaan 20.

$$\alpha_t = \alpha_0 * dr^{\frac{1+e}{ed}} \quad (20)$$

α_t : *learning rate baru*
α_0 : *learning rate sebelumnya*
d_r : *drop ratio.*
e : *epoch number*
ed : *determined interval epoch.*

2) Time-based decay: Penurunan laju *learning rate* secara berangsur-angsur berdasarkan waktu atau jumlah iterasi selama pelatihan, lihat persamaan 21.

$$\alpha_t = \frac{1}{1+d*e} * \alpha_0 \quad (21)$$

αt : *learning rate baru*
α0 : *learning rate sebelumnya*
d : *decay rate*
e : *epoch number*

3) Exponential weight decay: Metode ini mengaplikasikan laju perubahan terhadap model diturunkan secara eksponensial berdasarkan jumlah epoh, menggunakan persamaan 22.

$$\alpha_t = \alpha_0 * k^e \quad (22)$$

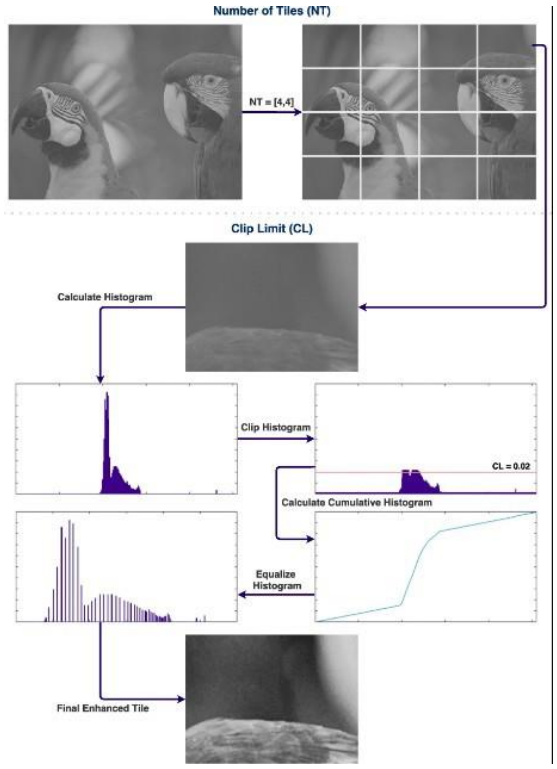
αt : *learning rate baru*
α0 : *learning rate sebelumnya*
k : *decay rate, biasanya < 1, paling umum 0.95*
e : *epoch number*

C. Augmentasi Citra

Augmentasi citra adalah teknik pengolahan citra yang digunakan dalam *deep learning* untuk menciptakan variasi baru dari data citra dengan melakukan transformasi atau manipulasi. Tujuan utama dari augmentasi citra adalah untuk meningkatkan keragaman data pelatihan sehingga model dapat belajar lebih baik, mengurangi risiko *overfitting*, dan meningkatkan performa model secara keseluruhan. Teknik augmentasi citra menggunakan fungsi transformasi geometri yaitu rotasi, *shifts*, *flips*, *brightness* dan *zoom*.

D. MCLAHE

Modified Contrast Limited Adaptive Histogram Equalization (MCLAHE) digunakan untuk meningkatkan kontras pada citra. MCLAHE merupakan modifikasi dari histogram *equalization* yang dapat mengatasi masalah munculnya *noise* pada citra dengan menerapkan dua langkah tambahan. Pertama, citra dibagi menjadi beberapa blok kecil dan lakukan histogram *equalization* pada setiap blok secara terpisah. Kedua, menetapkan batas kontras untuk mencegah amplifikasi yang berlebihan dan menjaga kualitas citra secara keseluruhan. Kemudian setiap blok digabungkan kembali, lihat Gambar 6.



Gambar. 6 CLAHE diagram[1]

II. HASIL DAN PEMBAHASAN

Tahapan identifikasi wajah terbagi atas tiga proses utama yaitu *preprocessing*, proses pelatihan, dan proses pengujian, lihat Gambar 7. Proses pelatihan dilakukan untuk mendapatkan model yang optimal sehingga menghasilkan akurasi yang tinggi. Proses pengujian dilakukan untuk menguji generalisasi model. Proses pengujian terdiri dari 2 tahapan, tahapan pertama adalah pengujian 2 model terbaik dengan metode *cross fold*

validation dengan nilai *fold* 10. Sementara pengujian kedua adalah pengukuran akurasi.

Dataset yang digunakan dalam penelitian ini ada dua yaitu *Cropped Extended* Yale *Face database* (Yale) dan dataset Komnet. Yale adalah dataset *frontal face*, terdapat 38 kelas dan masing-masing kelas terdiri dari 64 citra dengan beragam jenis iluminasi pencahayaan. Ukuran citra 168×192 piksel, format citra PNM (*Portable Any Map Image*).

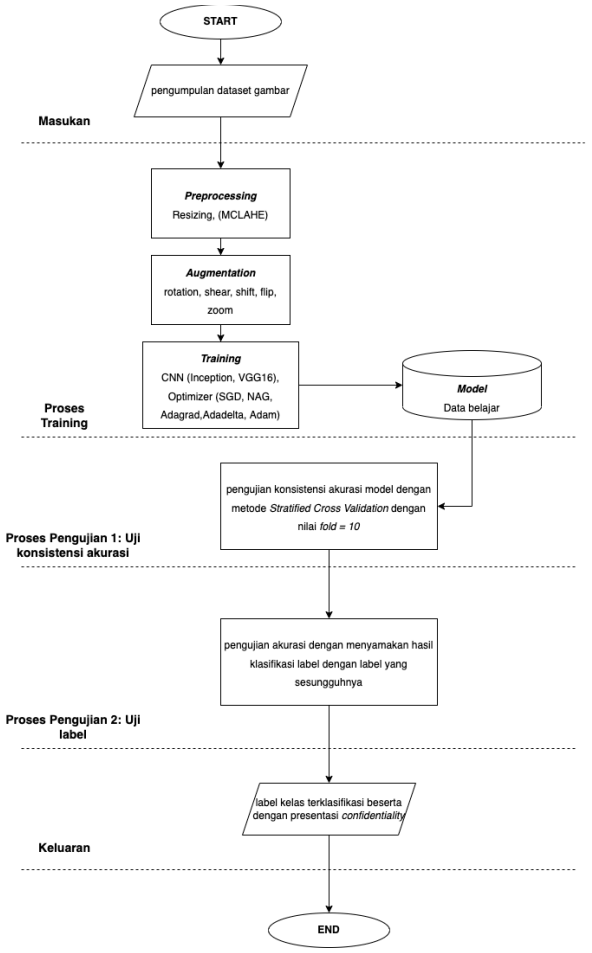
Dataset berikutnya adalah Komnet, dibuat oleh Laboratorium Politeknik Bali, terdiri 50 kelas dan masing masing kelas terdiri dari 24 citra wajah *frontal face*, berukuran 224 x 224 piksel, format jpeg. Citra di *capture* menggunakan media kamera digital, dikumpulkan dari sosial media, dan kamera *smartphone*.

Preprocessing dilakukan menerapkan MCLAHE pada dataset Yale dan KomNet, tujuannya adalah untuk menguji apakah pemrosesan MCLAHE berpengaruh terhadap meningkatnya akurasi model atau tidak.

Tahapan berikutnya augmentasi citra dengan menerapkan transformasi geometri rotasi +30°, *shifting* horisontal dan vertikal, *shearing* terhadap sumbu x dan y, *zoom* rentang 20%, -30%, dan *flip* horisontal. Tujuan transformasi adalah menambah variasi citra dari satu kelas yang sama sehingga mencegah model untuk mempelajari data latih terlalu detail sehingga tidak terjadi *overfitting*. Total citra yang digunakan pada penelitian, Yale dataset 1647 citra dan Komnet adalah 810. *Batch size* yang digunakan saat pelatihan adalah 32. Artinya akan ada 51 iterasi untuk Yale dan 25 iterasi untuk Komnet pada 1 epoh-nya.

Data augmentasi dibuat menggunakan ImageGenerator pustaka Keras. Jumlah data augmentasi setiap iterasi sebanyak *batch size*. Untuk data sampel Komnet diciptakan sebanyak 62500 data (25 x 25 x 100) dan *Yale* diciptakan sebanyak 260100 data (51 x 51 x 100), 100 menunjukkan jumlah epoh.

Bagian pelatihan menerapkan arsitektur VGG16 dan Inception. *Optimizer* merupakan *hyperparameter* yang akan diuji untuk menurunkan nilai *loss*. Pada penelitian kali ini 5 *optimizer* yang digunakan. Pengujian *hyperparameter* lainnya adalah *learning rate* dan perubahan nilai *learning rate* menggunakan *Learning Rate Scheduler*. Sistem pengenalan wajah merupakan klasifikasi multi kelas maka *cost function* yang dipakai adalah *Categorical Cross Entropy*. Pengukuran akurasi hasil pengujian menggunakan *Macro f1-score*. Pada sistem pengenalan wajah, *false positive* dan *false negative* lebih krusial dibandingkan dengan *true positive* dan *true negative* karena penting untuk mengetahui kesalahan prediksi, dibanding hanya mengetahui prediksi benar atau salah saja [18].

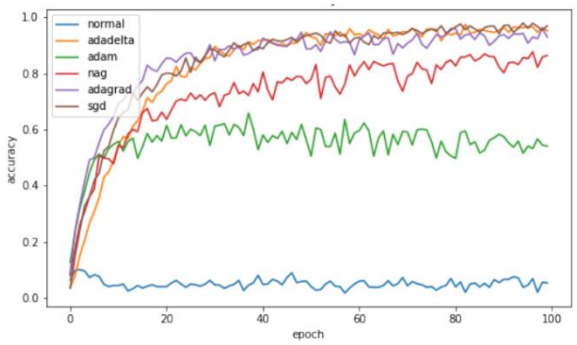


Gambar. 7 *Flowchart* global

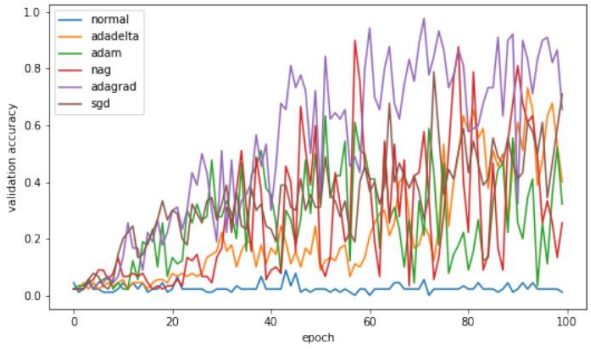
Dari seluruh dataset yang digunakan, kemudian dibagi dengan komposisi 75% untuk proses pelatihan dan 25% untuk proses pengujian. Pada proses pelatihan, dari 75% data yang sudah dibagi, diambil 10% digunakan sebagai data validasi proses pelatihan. Validasi ini diperlukan untuk melihat perkembangan model saat menerima data baru, apakah sudah memenuhi generalisasi dengan baik atau cenderung *overfit* atau *underfit*. Model dinyatakan *overfit* jika selisih akurasi dan akurasi validasi terlalu jauh, sedangkan model dinyatakan *underfit* jika akurasi terlalu rendah.

A. Pengujian Preprocessing

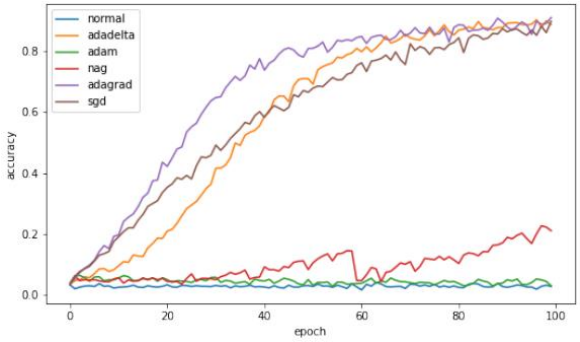
Berikut adalah grafik perbandingan akurasi pelatihan dan akurasi validasi untuk model VGG16 terhadap dataset Komnet. Gambar. 8 dan 9 dapat dilihat bahwa model tanpa MCLAHE cenderung mengalami *overfitting* sebab grafik akurasi validasi menunjukkan tingkat volatilitas yang ekstrim, kecenderungan yang sama juga terjadi untuk model Inception dan dataset Yale.



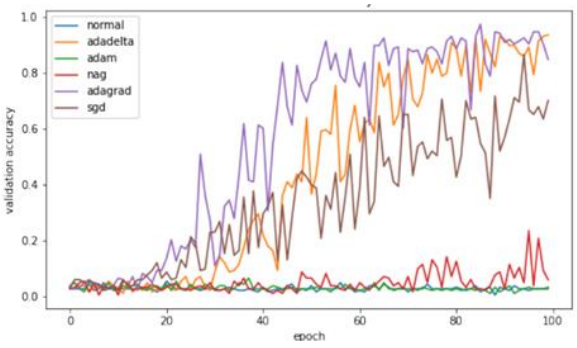
Gambar. 8 Akurasi VGG16+Origin Komnet



Gambar. 9 Akurasi validasiVGG16+Origin Komnet



Gambar. 10 Akurasi VGG16+ Komnet MCLAHE



Gambar. 11 Akurasi validasi VGG16+ Komnet MCLAHE

Grafik dalam Gambar. 10 dan 11, grafik masih menunjukkan *overfitting* tetapi memberikan dampak berkurangnya *overfitting* jika dibandingkan model tanpa MCLAHE. Kecenderungan yang sama untuk arsitektur Inception dan dataset Yale. Dari beragam pengujian memiliki pola yang sama maka dapat ditarik kesimpulan bahwa penggunaan *preprocessing* MCLAHE membantu dalam mengurangi tingkat *overfitting* pada model.

B. Pengujian Jenis Arsitektur dan Optimizer

Skenario pengujian untuk arsitektur VGG16 dan Inception adalah membuat kombinasi menggunakan *preprocessing* MCLAHE atau tidak untuk ke dua dataset, membuat kombinasi *hyperparameter* dengan diinisialisasi epoh 100, *learning rate* 0.01 dan 0.001, *batch size* 32, dan jenis *optimizer*, hasil Tabel 1.

Hasil pengujian menunjukkan, dataset Yale dengan model Inception secara keseluruhan memiliki *macro f1-score* lebih tinggi dibandingkan model VGG16. *Macro f1-score* tertinggi dicapai oleh model VGG16 dan *optimizer* Adagrad adalah 0.88. Model Inception dan *optimizer* NAG adalah 0.92. Hal ini disebabkan karena model Inception bekerja secara paralel dengan 3 ukuran kernel yang bervariasi sehingga mengekstraksi fitur lebih banyak dibandingkan dengan VGG16 yang menggunakan ukuran kernel yang sama yaitu 3 X 3.

Untuk dataset Yale+MCLAHE, *macro f1-score* tertinggi adalah arsitektur VGG16 dan Adadelta dengan *score* 0.90 sementara untuk arsitektur Inception *score* tertinggi adalah *optimizer* AdaDelta dengan nilai 0.97.

Secara keseluruhan grafik akurasi data uji Yale+MCLAHE menunjukkan memiliki tingkat volatilitas lebih rendah dibandingkan dengan data uji Yale. Disimpulkan bahwa metode MCLAHE ini membantu mengurangi tingkat *overfitting* suatu model.

Pada dataset Komnet untuk model VGG16 dan Inception, secara rata-rata *macro f1-score* mengalami peningkatan dibandingkan dengan dataset Yale. Nilai *macro f1-score* tertinggi yang dicapai oleh arsitektur VGG16 dan *optimizer* SGD 0.69, sementara untuk Inception dan *optimizer* AdaDelta mendapat skor tertinggi 1.00.

Dataset Komnet+MCLAHE untuk model Inception, *optimizer* Adadelta dan Adagrad mendapatkan *macro f1-score* pelatihan terbaik 0.98 sedangkan arsitektur VGG16 dengan *optimizer* AdaGrad adalah 0.94. Secara keseluruhan dataset Komnet+MCLAHE memberikan skor lebih baik, artinya *preprocessing* membantu dalam mengurangi tingkat volatilitas sehingga mengurangi tingkat *overfitting* pada model. Hasil semua pengujian yang dilakukan tanpa *optimizer* menunjukkan hasil yang tidak stabil dan disimpulkan mengalami *underfitting.*

TABEL I
F1-SCORE HASIL PENGUJIAN OPTIMIZER

Dataset	Optimizer	F1-score	
		VGG16	Inception
Yale	-	0.01	0.76
	SGD	0.73	0.81
	NAG	0.04	0.92
	Adagrad	0.88	0.63
	Adadelta	0.85	0.83
	Adam	0.80	0.70
Yale+MCLAHE	-	0.04	0.84
	SGD	0.83	0.75

Dataset	Optimizer	F1-score	
		VGG16	Inception
	NAG	0.01	0.94
	Adagrad	0.86	0.86
	Adadelta	0.90	0.97
	Adam	0.00	0.68
Komnet	-	0.01	0.27
	SGD	0.69	0.96
	NAG	0.19	0.99
	Adagrad	0.62	0.99
	Adadelta	0.44	1.00
	Adam	0.22	0.92
Komnet+MCLAHE	-	0.01	0.93
	SGD	0.74	0.97
	NAG	0.59	0.95
	Adagrad	0.94	0.98
	Adadelta	0.54	0.98
	Adam	0.22	0.47

Gambar. 12 Uji konsistensi VGG16

Gambar. 13 Uji konsistensi Inception

Pada Gambar 12, uji konsistensi dengan menggunakan *hyperparameter* terbaik, kurva pergerakan nilai akurasi dari model VGG16 selama 10 kali pelatihan dengan metode *Stratified Cross Validation*. Akurasi yang diperoleh pada setiap tahapannya adalah sebagai berikut: 0.93, 0.967, 0.9416, 0.9, 0.941, 0.967, 0.925, 0.941, 0.93, 0.967. Rata-rata akurasi dari *cross validation* adalah 0.9417. Kurva pergerakan nilai akurasi dari model Inception selama 10 kali pelatihan, akurasi yang diperoleh pada setiap tahapannya adalah sebagai berikut: 0.975, 1.00, 0.983, 0.99, 0.99, 1.00, 1.00, 1.00, 0.99, 1.00. Rata-rata akurasi yang diperoleh adalah 0.993, lihat Gambar 13.

C. Pengujian Learning Rate

Penelitian ini tidak hanya berfokus pada pengaruh *optimizer*, jenis arsitektur, serta *preprocessing* MCLAHE terhadap akurasi model tetapi juga menguji pengaruh *learning rate tuning*. Perubahan *learning rate* yang diuji adalah *Step Decay* (SD), *Time Based Decay* (TBD), dan *Exponential Weight Decay* (EWD), dan tanpa menggunakan *scheduler* (No LR). Pengujian dilakukan untuk masing-masing dataset, model arsitektur, dan *optimizer*. Tabel 2 menampilkan hasil pengujian *learning rate* untuk model dengan akurasi terbaik dataset Komnet+MCLAHE dan Inception.

TABEL II
PENGUJIAN LEARNING RATE TUNING KOMNET MCLAHE DENGAN MODEL INCEPTION

OPTIMIZER	NO LR	SD	TBD	EWD
Normal	0.93	0.25	-	-
SGD	0.97	0.06	0.69	0.00
NAG	0.95	0.26	0.02	0.00
Adagrad	0.98	0.11	0.96	0.00
Adadelta	0.98	0.01	0.65	0.00
Adam	0.47	0.93	0.35	0.00

Hasil pengujian untuk dataset Komnet+MCLAHE, disimpulkan bahwa peningkatan skor hanya terjadi saat menggunakan *Step Decay* (SD) untuk *optimizer* Adam, jika tanpa *Scheduler* (No LR) mendapatkan skor 0.47, terdapat peningkatan skor menjadi 0.93. Pengujian akurasi tanpa *optimizer* (Normal) memiliki kecenderungan akurasi naik pada *Step Decay* untuk semua model VGG16 sedangkan model Inception hanya pada dataset origin Komnet.

IV. KESIMPULAN

Penggunaan *preprocessing* MCLAHE memberikan hasil akurasi yang lebih stabil dan tidak fluktuatif untuk model VGG16 dan Inception di kedua dataset. Untuk akurasi model disimpulkan bahwa proses MCLAHE meningkatkan akurasi model terjadi pada skenario yaitu, dataset Yale+MCLAHE untuk kedua model mendapatkan *macro f1-score* tertinggi pada *optimizer* AdaDelta, untuk model Inception skor 0.97 dan VGG 0.90. Dataset Komnet+MCLAHE mendapatkan *macro f1-score* tertinggi pada model VGG16 *optimizer* AdaGrad 0.94 sedangkan Inception tertinggi adalah pada *optimizer* AdaGrad dan AdaDelta 0.98. Secara keseluruhan hasil pengujian akurasi, model Yale+MCLAHE dan Komnet+MCLAHE memiliki tingkat volatilitas lebih rendah dibandingkan dengan dataset origin. Disimpulkan bahwa metode MCLAHE ini membantu mengurangi tingkat *overfitting* suatu model.

Akurasi rata-rata tertinggi diperoleh pada proses pengujian model Inception dengan *optimizer* Adadelta. Hal ini disebabkan Inception mendukung proses ekstraksi fitur yang kompleks dengan memanfaatkan proses konvolusi paralel dengan 3 dimensi kernel/ *filter* sekaligus sehingga fitur hasil ekstraksi lebih beragam. Selain itu algoritma

optimizer AdaDelta lebih efektif mengubah nilai *learning rate* untuk meningkatkan perubahan nilai bobot dan bias.

Penggunaan algoritma LRS mempengaruhi tingkat akurasi model. Dari 48 total pengujian yang dilakukan, terdapat 28 kasus uji (58.3%) yang mengalami peningkatan akurasi akibat penggunaan nilai *learning rate* yang dinamis. Berikut pasangan *optimizer* beserta LRS yang memberikan peningkatkan akurasi.

- *Optimizer* Adam: *Step Decay*
- *Optimizer* SGD, NAG, dan Adagrad: *Time based decay*

REFERENSI

[1] R. I. Bendjilali, B. Mohammed, M. Khaled, and A. Taleb-Ahmed, Illumination-robust face recognition based on deep convolutional neural networks architecture, Indonesian Journal of Electrical Engineering and Computer Science, May 2020.

[2] J. Brownlee, "Why Optimization Is Important in Machine," Machine Learning Mastery, [Online] https://machinelearningmastery.com/. [Accessed: February 12, 2022].

[3] M. Yaqub, J. Feng, S. Zia, A. Arshid, K. Jia, U. R. Zaka, and A. Mehmood, "State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images," in Brain Sciences, published July 3, 2020.

[4] S. S. Mohammed, W. A. Mohamed, A.T. Khalil, and A.S. Mora, "Deep Learning Face Detection and Recognition," in International journal of electronics and telecommunications, June 2019.

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

[6] M. Coskun, A. Ucar, O. Yildrim, and Y. Demir, "Face Recognition Based on Convolutional Neural Network," in Conference Paper, November 2021.

[7] Z. Xie, J. Li, and H. Shi, "A Face Recognition Method Based on CNN," in Journal of Physics: Conference Series, 2019, doi: 10.1088/1742-6596/1395/1/012006.

[8] M. K. Hasan, M. S. Ahsan, A. A. Mamun, S. H. S. Newaz, and G. M. Lee, "Human Face Detection Techniques: A Comprehensive Review and Future Research Directions," Electronics 2021, 10, 2354, [Online] https://doi.org/10.3390/electronics10192354.

[9] L. Patil and V.D. Mytri, "Face Recognition with CNN and Inception Deep Learning Models," International Journal of Recent Technology and Engineering (IJRTE), Volume-8 Issue-3, September 2019.

[10] H. E. Khiyari and H. Wechsler, "Face Recognition across Time Lapse Using Convolutional Neural Networks," in Journal of Information Security, Vol.7 No.3, 2016.

[11] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in The 3rd International Conference on Learning Representations ICLR, 2015, [Online] https://arxiv.org/abs/1409.1556. [Accessed: January 20, 2022].

[12] A. Rosebrock, Deep Learning for Computer Vision with Python, PyImageSearch. 2017.

[13] M. Davidson, "Review on Gradient Descent Algorithms in Machine Learning," in Journal for Innovative Development in Pharmaceutical and Technical Science, Volume-2, Issue-10, October 2019, [Online] https://ssrn.com/abstract=3810947.

[14] S. Ruder, "An overview of gradient descent optimization algorithms", [Online] https://www.semanticscholar.org/reader/769ef3d5021cd71c37d2c403f231a53d1accf786. [Accessed: February 1, 2022, 16:28:00].

[15] Vidushi et al., "Assessment of Optimizers impact on Image Recognition with Convolutional Neural Network to Adversarial Datasets," in Journal of Physics: Conference Series, 2021, doi:10.1088/1742-6596/1998/1/012008.

[16] K. Naik, "Understanding All Optimizers In Deep Learning," [Online] https://krishnaik.in/2022/03/28/understanding-all-optimizers-in-deep-learning. [Accessed: February 7, 2022, 23:41:00].

[17] Y. Ding, "The Impact of Learning Rate Decay and Periodical Learning Rate Restart on Artificial Neural Network," in Proceedings of the 2021 2nd International Conference on Artificial Intelligence in Electronics Engineering, pp. 6–14, 2012, [Online] https://doi.org/10.1145/3460268.3460270.

[18] P. Huilgol, "Accuracy vs F1-Score," [Online] https://medium.com/analytics-vidhya/. [Accessed: 12 February 2022, 22:39:00].