Perbandingan AlexNet dan VGG untuk Pengenalan Ekspresi Wajah pada Dataset Kelas Komputasi Lanjut

Comparison of AlexNet and VGG for Facial Expression Recognition on Advanced Computing Class Dataset

Sri Nurdiati[1], Mohamad Khoirun Najib[2], Fahren Bukhari[3], Muhammad Reza Ardhana[4], Salsabilla Rahmah[5], Trianty Putri Blante[6]
[1,2,3,4,5,6]Program Studi S2 Matematika Terapan, IPB University
Jl. Meranti Kampus, Babakan, Kec. Dramaga, Bogor, Jawa Barat 16680
E-mail: [1]nurdiati@apps.ipb.ac.id , [2]mkhoirun_najib@apps.ipb.ac.id , [3]fahrenbu@apps.ac.id , [4]ardhana_reza17@apps.ipb.ac.id , [5]2502salsabilla@apps.ipb.ac.id , [6]putriblante@apps.ac.id

Abstrak

Pengenalan emosi memainkan peran penting dalam komunikasi yang dapat dikenali dari ekspresi wajah. Terdapat banyak metode yang dapat digunakan untuk mengenali ekspresi wajah secara automatis, seperti *convolutional neural network* (CNN). Penelitian ini bertujuan untuk mengimplementasikan dan membandingkan model CNN dengan arsitektur AlexNet dan VGG untuk pengenalan ekspresi wajah menggunakan bahasa pemrograman Julia. Model CNN akan digunakan untuk mengklasifikasikan tiga ekspresi yang berbeda dari tujuh orang pengekspresi. Data diproses dengan beberapa teknik augmentasi data untuk mengatasi masalah keterbatasan data. Hasil penelitian menunjukkan bahwa ketiga arsitektur dapat mengklasifikasikan ekspresi wajah dengan sangat baik, yang ditunjukkan oleh nilai rata-rata akurasi pada data *training* dan *testing* yang lebih dari 94%. Untuk memenuhi nilai *cross-entropy* sebesar 0.1, arsitektur VGG-11 memerlukan jumlah *epoch* yang paling sedikit dibandingkan arsitektur lainnya, sedangkan arsitektur AlexNet memerlukan waktu komputasi yang paling sedikit. Waktu komputasi pada proses pelatihan sebanding dengan jumlah parameter yang terkandung pada model CNN. Akan tetapi, jumlah *epoch* yang sedikit belum tentu membutuhkan waktu komputasi yang sedikit. Selain itu, nilai *recall*, presisi, dan F1-*score* untuk masalah klasifikasi *multi-class* menunjukkan hasil yang baik, yaitu lebih dari 94%.

Kata kunci: Adam, *Confusion Matrix*, Pelatihan Fitur, Sensitivitas, Tingkat Positif Benar

Abstract

Emotion recognition plays an important role in communication that can be recognized from facial expressions. Many methods can be used to automatically recognize facial expressions, such as a convolutional neural network (CNN). This study aims to implement and compare a CNN model with AlexNet and VGG architectures for facial expression recognition problems using Julia. The CNN model will classify three different expressions from seven people. The data is processed with several data augmentations to overcome the limitations. The results show that the three architectures can classify facial expressions very well, based on the average value of accuracy on the training and testing data which is more than 94%. To meet the cross-entropy value of 0.1, the VGG-11 architecture requires the least number of epochs compared to other architectures, while the AlexNet architecture requires the least computational time. The computational time in the training process is proportional to the number of parameters contained in the CNN model. However, a small number of epochs does not require less computational time. Moreover, the recall, precision, and F1-score values for multi-class classification problems show good results, which are more than 94%.

Keywords: Adam, Confusion Matrix, Feature Learning, Sensitivity, True Positive Rate

1.PENDAHULUAN

Pengenalan emosi memainkan peran penting dalam komunikasi dan masih merupakan bidang penelitian yang menantang [1]. Emosi tersebut dapat dikenali dari ekspresi wajah yang menjadi indikator perasaan seseorang [2]. Oleh karena itu, keadaan emosi seseorang dapat segera dikenali berdasarkan ekspresi wajahnya. Pada era digital seperti sekarang, pengenalan emosi tidak hanya diperlukan oleh interaksi antar manusia, tetapi interaksi manusia dengan mesin juga memerlukannya. Sebagai contoh, aplikasi dunia nyata yang melibatkan interaksi tersebut seperti, interaksi robot-manusia [3], *virtual reality* [4], *augmented reality* [5], hingga sistem asisten pengemudi canggih [6]. Dengan perkembangan teknologi yang sangat pesat, masalah sistem automatis, seperti pengenalan emosi seseorang dari gambar wajah menjadi mudah untuk dipecahkan. Sebuah mesin dapat secara akurat berinteraksi dengan manusia jika memiliki kemampuan untuk mengenali emosi manusia.

Untuk sistem pengenalan ekspresi wajah automatis, wajah dideteksi dan direkam sebagai citra dua dimensi menggunakan perangkat, seperti kamera. Suatu sistem pengenal ekspresi wajah automatis biasanya terdiri dari empat tahap, yaitu tahap pra-pemrosesan, ekstraksi fitur, pemilihan fitur, dan klasifikasi ekspresi wajah [7]. Pada tahap pra-pemrosesan, area wajah pertama kali dideteksi dan diekstraksi dari citra *input* karena merupakan area yang berisi informasi terkait ekspresi. Selanjutnya, ekstraksi fitur akan membedakan fitur-fitur yang terdapat pada citra wajah, seperti mata, alis, hidung, mulut dan area wajah lainnya. Fitur-fitur tersebut dipilih pada tahap pemilihan fitur, sehingga diperoleh beberapa fitur yang mengandung lebih banyak informasi untuk mengklasifikasikan kelas yang berbeda. Pada langkah terakhir (klasifikasi ekspresi wajah), metode pengklasifikasian berbasis *machine learning* seperti *k-nearest neighbor* [8], *support vector machine* [9], dan *neural network* [10] terlebih dulu dilatih, kemudian digunakan untuk mengklasifikasikan ekspresi wajah.

Pada akhir-akhir ini, *deep learning* banyak digunakan pada berbagai tugas *computer vision* termasuk pengenalan ekspresi wajah dan mencapai keberhasilan yang sangat baik [11]. Salah satu teknik dalam *deep learning* yang sering digunakan untuk masalah pengklasifikasian citra, termasuk pengenalan ekspresi wajah, yaitu *convolutional neural network* (CNN). Secara umum, struktur CNN memiliki dua proses, yaitu pembelajaran fitur (*feature learning*) dan klasifikasi. Dalam CNN, tahap ekstraksi dan pemilihan fitur pada suatu sistem pengenalan ekspresi wajah dilakukan secara automatis pada tahap *feature learning*. Hasil *feature learning* kemudian diklasifikasikan menggunakan *fully-connected layer*.

Karena keberhasilan yang baik dari teknik CNN dalam menyelesaikan masalah *computer vision*, berbagai macam arsitektur CNN dieksplorasi untuk mencapai klasifikasi yang lebih baik [12]. Arsitektur CNN yang pertama kali terkenal adalah LeNet-5 [13], yang digunakan untuk pengenalan angka tulisan tangan. LeNet-5 menjadi salah satu promotor dalam perkembangan *deep learning* hingga saat ini. Pada tahun 2012, varian dari LeNet-5 yaitu AlexNet, memenangkan tantangan klasifikasi ImageNet dengan kesalahan di sekitar 16%. Setelah itu, variasi lain dari LeNet-5, yaitu VGG-16, menjadi *runner-up* tantangan tersebut pada tahun 2014 dengan kesalahan di sekitar 7.4% [14]. Masalah utama klasifikasi menggunakan CNN adalah jumlah parameternya yang sangat banyak, sehingga proses pelatihannya membutuhkan waktu yang cukup lama [15].

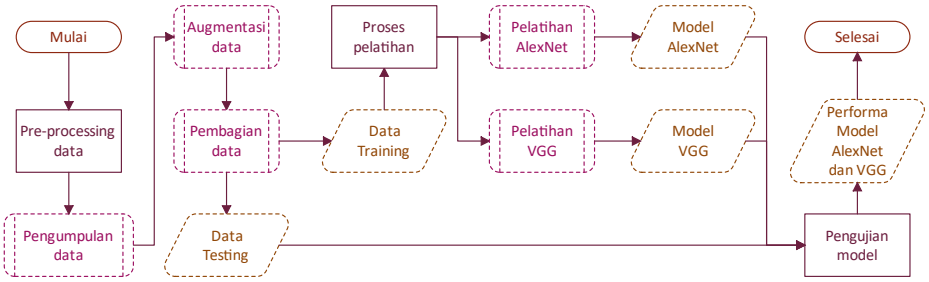
Berdasarkan uraian tersebut, penelitian ini bertujuan untuk mengimplementasikan dan membandingkan arsitektur pada CNN, yaitu AlexNet dan VGG, untuk masalah pengenalan ekspresi wajah. Proses komputasi dikerjakan menggunakan bahasa pemrograman Julia yang telah diuji memiliki performa yang lebih cepat dibandingkan bahasa pemrograman dinamis lain, seperti Python, R, dan Matlab [16], [17].

2. METODE PENELITIAN

Penelitian ini diawali dengan pengumpulan data ekspresi wajah yang berasal dari tujuh orang mahasiswa program studi S-2 Matematika Terapan, IPB University angkatan 57 yang mengambil mata kuliah komputasi lanjut pada tahun 2022. Masing-masing mahasiswa memberikan tiga jenis ekspresi wajah, yaitu netral, senyum dan seringai, sehingga jumlah data awal yang diperoleh adalah 27 citra wajah. Karena CNN membutuhkan data yang cukup banyak untuk proses pelatihannya, data awal yang diperoleh tersebut diperbanyak menggunakan teknik augmentasi data.

Augmentasi data adalah suatu metode yang digunakan untuk mengatasi masalah keterbatasan data pada *machine learning*. Augmentasi mencakup serangkaian teknik untuk meningkatkan ukuran dan kualitas data *training*, sehingga model *machine learning* dapat bekerja lebih baik [18]. Adapun prosedur augmentasi data yang digunakan pada penelitian ini untuk meningkatkan kualitas dan memperbanyak data [19] adalah 1) skala: muka asli, diperlebar dan diperkecil, 2) kecerahan: -20%, 0%, dan +20%, dan 3) kontras: -40%, 0%, dan +40%. Dengan augmentasi data tersebut, satu citra dapat diperbanyak menjadi 27 citra. Jadi, total keseluruhan data yang diperoleh adalah 567 citra, yang kemudian dibagi ke dalam dua bagian, yaitu data *training* (80%) dan *testing* (20%).

Penelitian ini menggunakan model CNN dengan arsitektur AlexNet dan VGG untuk mengklasifikasikan ekspresi wajah. Model tersebut dilatih menggunakan data *training*. Setelah itu, model diuji menggunakan data *testing* untuk mengetahui performa kedua model. Diagram alur pada penelitian ini dapat dilihat pada Gambar 1.



Gambar 1 Diagram alir penelitian

2.1 Convolutional Neural Network (CNN)

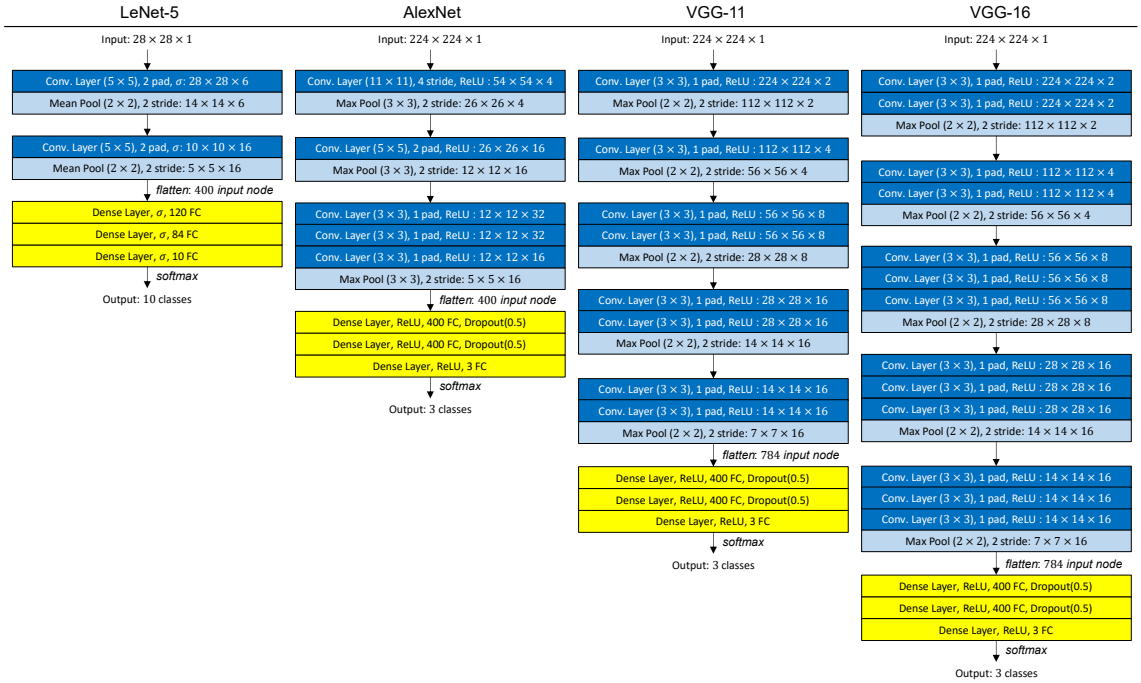
CNN adalah arsitektur *deep learning* yang terinspirasi oleh mekanisme persepsi visual alami pada makhluk hidup [20]. Arsitektur CNN terdiri dari kombinasi tiga jenis lapisan (*layer*), yaitu *convolution*, *sub-sampling*, dan *classification layer*. Proses pertama model CNN adalah *feature learning* yang terdiri atas *convolution* dan *sub-sampling layer*. Node hasil dari *convolution* dan *sub-sampling layer* dikelompokkan ke dalam bidang dua dimensi yang disebut *feature map*. Setiap bidang *layer* diturunkan dari kombinasi satu atau lebih bidang pada *layer* sebelumnya. Setiap node dari *convolution layer* mengekstrak fitur dari citra *input* dengan operasi konvolusi.

Saat fitur menyebar ke layer yang lebih tinggi, dimensi fitur berkurang tergantung pada ukuran kernel yang digunakan untuk operasi pada *convolution* dan *sub-sampling layer*. Namun, jumlah *feature maps* biasanya meningkat untuk mewakili fitur yang lebih baik dari gambar *input* guna memastikan akurasi klasifikasi. Hasil pada lapisan terakhir *feature learning* kemudian diratakan dan digunakan sebagai data *input* ke dalam *fully-connected layer* yang disebut *classification layer*. Skor dari setiap kelas dihitung pada *classification layer* terakhir dengan *softmax layer*. Model CNN memberikan *output* untuk kelas yang sesuai berdasarkan skor yang tertinggi.

2.2 Arsitektur LeNet-5, AlexNet dan VGG

LeNet-5 memiliki kemampuan yang sangat baik dalam kasus klasifikasi gambar dengan resolusi yang rendah. Pada dasarnya, LeNet-5 diciptakan untuk mengklasifikasikan gambar tulisan tangan yang berisi angka 0-9. Resolusi *input* yang dibutuhkan pada arsitektur LeNet-5 adalah 28×28 atau bisa juga pada resolusi 32×32. Hal ini menjadi kurang relevan dalam kasus klasifikasi ekspresi wajah, karena gambar dengan resolusi rendah tersebut tidak mampu menangkap secara detail raut muka seseorang. Dengan demikian, pengenalan ekspresi wajah akan lebih sesuai apabila menggunakan arsitektur yang memiliki *input* dengan resolusi yang lebih tinggi.

Arsitektur AlexNet dan VGG adalah variasi dari arsitektur LeNet-5 yang memiliki resolusi *input* sebesar 224×224. Resolusi tersebut cukup untuk menangkap detail raut muka seseorang. Namun, kedua arsitektur tersebut pada awalnya diciptakan untuk mengklasifikasikan tantangan ImageNet yang terdiri dari 10 ribu kelas citra yang berbeda. Oleh karena itu, arsitektur AlexNet dan VGG terlalu kompleks untuk diterapkan pada kasus klasifikasi ekspresi wajah, yang memiliki tiga kelas citra. Untuk mengatasi hal tersebut, penelitian ini akan melakukan penyesuaian arsitektur AlexNet dan VGG dengan cara mengurangi jumlah *feature maps* (*channel*) yang dihasilkan di setiap *convolution layer*. Perbandingan arsitektur LeNet-5 serta arsitektur AlexNet dan VGG dapat dilihat pada Gambar 2.



Gambar 2 Perbandingan model *convolutional neural network* arsitektur LeNet-5 dengan arsitektur AlexNet, VGG-11, dan VGG-16 yang telah disesuaikan jumlah *feature maps* masing-masing lapisan.

Angka 5 pada LeNet-5 merujuk pada jumlah processing layer pada arsitektur tersebut, yaitu dua *convolution layer* dan tiga *fully-connected layer*. Ciri khas dari arsitektur LeNet-5 ini adalah adanya dua pasangan *convolution* dan *subsampling layer* pada proses *feature selection*. *Sub-sampling layer* yang digunakan adalah *average pooling*. Setiap *processing layer* menerapkan fungsi sigmoid sebagai fungsi aktivasi, yang diberikan oleh

$$f_\sigma(x) = (1 + e^{-x})^{-1} \quad (1)$$

Arsitektur ini diakhiri dengan fungsi *softmax* untuk merepresentasikan nilai *output* di suatu kelas sebagai persentase relatif terhadap kelas lainnya, yaitu

$$\sigma_M(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad (2)$$

dengan z adalah vektor yang berisi nilai dari setiap kelas dan z_i adalah nilai pada kelas ke-i. Fungsi *softmax* ini juga dikenal sebagai *softargmax* atau eksponensial ternormalisasi adalah generalisasi dari fungsi sigmoid pada dimensi yang lebih banyak.

Sedikit berbeda dengan LeNet-5, arsitektur AlexNet memiliki delapan *processing layer*. Ciri khas dari AlexNet adalah adanya tiga *convolution layer* yang berurutan setelah dua pasang *convolution* dan *subsampling layer* pada proses *feature selection*. *Sub-sampling layer* yang digunakan pada AlexNet adalah *max pooling*. Arsitektur asli dari AlexNet menghasilkan sebanyak 256 *feature maps* pada akhir proses *feature learning*. Namun, penelitian ini melakukan sedikit penyesuaian pada *feature maps* yang dihasilkan sehingga jumlah *feature maps* yang dihasilkan dapat direduksi menjadi hanya 16 *feature maps*. Setiap *processing layer* pada AlexNet menerapkan fungsi *rectified linear unit* (ReLU) sebagai fungsi aktivasi, yang diberikan oleh

$$f_{ReLU}(x) = x^+ = \max(0, x) \tag{3}$$

Fungsi ini juga dikenal sebagai fungsi *ramp*.

Sementara itu, VGG merupakan varian lain dari LeNet dengan *feature learning* yang lebih dalam daripada AlexNet. Berdasarkan jumlah *processing layer*-nya, terdapat beberapa macam variasi VGG, seperti VGG-11 yang merujuk pada 11 *processing layer* yang digunakan, VGG-13, VGG-16, dan VGG-19. Penelitian ini akan menggunakan dua variasi VGG, yaitu VGG-11 dan VGG-16 dengan penyesuaian seperti pada AlexNet, yaitu berupa reduksi pada jumlah *feature maps*. Sama seperti AlexNet, arsitektur VGG menggunakan *max pooling* untuk *sub-sampling layer* dan ReLU sebagai fungsi aktivasi.

2.3 Adam Optimizer

Model CNN dengan arsitektur AlexNet, VGG-11, dan VGG-16 dilatih menggunakan Adam, yang berasal dari kata *adaptive moment estimation*. Adam merupakan sebuah metode untuk optimasi stokastik efisien yang hanya membutuhkan gradien pada orde pertama dengan sedikit kebutuhan memori. Adam dirancang untuk menggabungkan keuntungan dari dua metode, yaitu AdaGrad [21] dan RMSProp [22]. Beberapa keunggulan dari Adam adalah proses komputasi efisien, persyaratan memori kecil, invarian terhadap skala ulang diagonal dari gradien, sangat cocok untuk masalah yang besar dalam hal data dan/atau parameter, cocok untuk tujuan non-stasioner, dan cocok untuk masalah dengan gradien yang sangat bising (*noisy*) atau jarang (*sparse*). Algoritma 1 menunjukkan *pseudocode* dari Adam optimizer [23].

Algoritma 1. *Adam Optimizer*

required: η (ukuran langkah atau *learning rate*)
required: $\beta_1, \beta_2 \in [0,1)$ (Laju peluruhan eksponensial)
required: $f(\theta)$ (Fungsi tujuan stokastik dengan parameter θ)
required: θ_0 (Inisiasi vektor parameter)
$m_0 \leftarrow 0$ (Inisialisasi vektor momen pertama)
$v_0 \leftarrow 0$ (Inisialisasi vektor momen kedua)
$t \leftarrow 0$ (Inisialisasi langkah iterasi)
while θ_t belum konvergen ***do***
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Hitung gradien mengacu pada tujuan stokastik)
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Perbarui estimasi momen pertama)
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Perbarui estimasi momen kedua)
  $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Hitung estimasi momen pertama terkoreksi bias)
  $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Hitung estimasi momen kedua terkoreksi bias)
  $\theta_t \leftarrow \theta_{t-1} - \eta \cdot \hat{m}_t / \left(\sqrt{\hat{v}_t} + \epsilon\right)$ (Perbarui parameter)
end while
return θ_t

Beberapa nilai *hyper-parameter* perlu didefinisikan sebelum menggunakan Adam *optimizer*. Menurut Kingma & Ba [23], pengaturan yang baik untuk masalah *deep learning*

adalah $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, dan $\epsilon = 10^{-8}$. Selain itu, fungsi tujuan stokastik yang digunakan adalah *cross-entropy*, yang diberikan oleh

$$f_{CE}(y, \hat{y}) = -\frac{1}{k}\sum_{i=1}^{k} y_i \cdot \log \hat{y}_i \tag{4}$$

dengan k menunjukkan banyaknya kelas, y dan $\hat{y}$ masing-masing menunjukkan vektor yang berisi peluang persentase setiap kelas pada label aktual dan prediksi dari model CNN. Setiap model CNN akan dilatih maksimal sebanyak 1000 *epochs* atau nilai fungsi *cross-entropy* bernilai kurang dari 0.1.

2.4 Ukuran Performa Model

Setelah model dilatih, performa dari model tersebut dinilai menggunakan *confusion matrix*, berdasarkan data *training* dan *testing*. Untuk kasus klasifikasi tiga kelas, *confusion matrix* yang diperoleh dapat dilihat pada Gambar 3. Nilai $C_{1,1}$ menunjukkan banyaknya data yang sebenarnya kelas 1 diprediksi oleh model pada kelas 1 juga, sedangkan $C_{1,2}$ menunjukkan data yang sebenarnya adalah kelas 1, tetapi diprediksi kelas 2 oleh model. Interpretasi yang sama juga berlaku untuk nilai $C_{1,3}$, $C_{2,1}$, hingga $C_{3,3}$.

		Prediksi		
		Kelas 1	Kelas 2	Kelas 3
Aktual	Kelas 1	$C_{1,1}$	$C_{1,2}$	$C_{1,3}$
	Kelas 2	$C_{2,1}$	$C_{2,2}$	$C_{2,3}$
	Kelas 3	$C_{3,1}$	$C_{3,2}$	$C_{3,3}$

Gambar 3 *Confusion matrix* pada klasifikasi tiga kelas berbeda

Berdasarkan *confusion matrix* yang dihasilkan, nilai akurasi model prediksi dapat dihitung. Untuk kasus klasifikasi dengan n kelas, akurasi ditentukan menggunakan persamaan berikut.

$$Accuracy = \frac{\sum_{i=1}^{n} C_{i,i}}{\sum_{j=1}^{n}\sum_{i=1}^{n} C_{i,j}} \tag{5}$$

Nilai akurasi tersebut merupakan rasio antara jumlah diagonal utama (*trace*) dan jumlah semua titik pada *confusion matrix*.

Selain akurasi, ukuran lainnya seperti *recall*, presisi, dan F1-*score* untuk masing-masing kelas dapat diperoleh juga melalui *confusion matrix*, yaitu

$$TPR(C_i) = \frac{C_{i,i}}{\sum_{j=1}^{n} C_{i,j}} \tag{6}$$

$$PPV(C_i) = \frac{C_{i,i}}{\sum_{j=1}^{n} C_{j,i}} \tag{7}$$

$$F_1(C_i) = 2 \cdot \frac{TPR(C_i) \cdot PPV(C_i)}{TPR(C_i) + PPV(C_i)} \tag{8}$$

dengan $i = 1,2,\dots,n$ [24]. Nilai $TPR(C_i)$ menunjukkan nilai *true positive rate* (*recall*), nilai $PPV(C_i)$ menunjukkan nilai *positive predictive value* (presisi), dan nilai $F_1(C_i)$ menunjukkan F1-*score* masing-masing untuk kelas i. Berdasarkan persamaan tersebut, setiap kelas memiliki nilai *recall*, presisi, dan F1-*score* masing-masing. Untuk menilai performa model secara lebih umum, nilai *recall*, presisi, dan F1-*score* dapat dihitung menggunakan pendekatan makro, yaitu

$$TPR(makro) = \frac{1}{n}\sum_{i=1}^{n} TPR(C_i) \tag{9}$$

$$PPV(makro) = \frac{1}{n}\sum_{i=1}^{n} PPV(C_i) \quad (10)$$

$$F_1(makro) = 2 \cdot \frac{TPR(makro) \cdot PPV(makro)}{TPR(makro) + PPV(makro)} \quad (11)$$

dengan $TPR(makro)$, $PPV(makro)$, dan $F_1(makro)$ adalah ukuran untuk menilai *recall*, presisi, dan F1-*score* model prediksi dengan pendekatan makro.

3. HASIL DAN PEMBAHASAN

Seluruh proses komputasi pada penelitian ini dikerjakan menggunakan laptop Lenovo tipe ideapad 330-14AST dengan prosesor AMD A4-9125 Radeon R3, 4 Core (2C+2G) 2.30 GHz dan RAM 8 GB. Penelitian ini menggunakan bahasa pemrograman Julia versi 1.6.5, yang memiliki performa cepat, bahasa yang dinamis, dan *open source*. Tahap *pre-processing* dilakukan menggunakan paket *Images.jl*, Sementara itu, pembentukan dan pelatihan model CNN menggunakan paket *Flux.jl* yang tersedia pada Julia.

3.1 Pre-processing Data

Berdasarkan tahapan penelitian yang ditunjukkan pada Gambar 1, tahap pertama penelitian ini adalah *pre-processing data*. Setiap citra yang telah diperoleh diekstraksi ke dalam representasi matriks, yaitu suatu matriks 3-dimensi $I_c(w,h,c)$, dengan masing-masing dimensi menunjukkan piksel lebar, tinggi, dan *channel* warna. Setelah itu, gambar dikonversi dari gambar berwarna (RGB) I_c menjadi hitam-putih (*grayscale*) I_g menggunakan persamaan berikut.

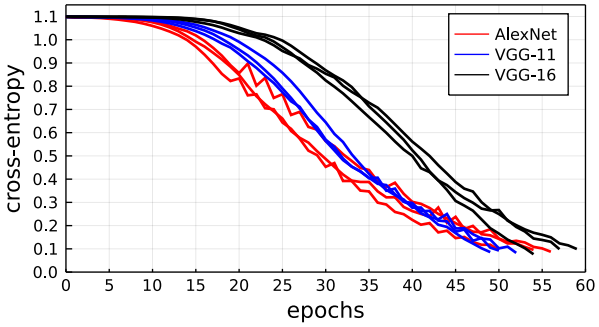
$$I_g(w,h) = 0.299 \cdot I_c(w,h,r) + 0.587 \cdot I_c(w,h,g) + 0.114 \cdot I_c(w,h,b) \quad (12)$$

dengan $I_c(w,h,r)$, $I_c(w,h,g)$ dan $I_c(w,h,b)$ merupakan matriks yang merepresentasikan spektrum warna merah, biru dan hijau pada matriks $I_c(w,h,c)$. Citra tersebut kemudian diaugmentasi dengan skenario yang telah ditetapkan dan data citra disimpan dalam urutan WHCN, yaitu *width* (lebar), *height* (tinggi), *channel*, dan ukuran *batch*. Dari proses ini, dihasilkan matriks dengan ukuran 224×224×1×567.

Setelah matriks citra diperoleh, matriks dipartisi menjadi dua bagian untuk proses pelatihan (*training*) dan pengujian (*testing*). Untuk kesetaraan supaya data *training* dan *testing* tidak selalu berubah, pembagian data dilakukan secara acak dengan menetapkan benih atau *seed* dari *random generator* yaitu 1. Dengan rasio pembagian 80:20%, diperoleh data *training* dan *testing* masing-masing sebanyak 454 dan 113 citra.

3.2 Pelatihan dan Pengujian Model

Setelah data selesai disiapkan, model CNN dengan arsitektur AlexNet, VGG-11, dan VGG-16 yang telah disesuaikan jumlah *feature maps* pada masing-masing lapisan dilatih menggunakan data *training*. Nilai inisiasi awal dari parameter pada setiap awal ditetapkan secara acak. Proses pelatihan dilakukan masing-masing sebanyak tiga kali untuk memenuhi nilai *loss function* (*cross-entropy*) 0.1 dengan menghitung jumlah *epoch* dan waktu komputasi. Perbandingan perambatan nilai *cross-entropy* untuk setiap arsitektur model CNN pada data *training* selama proses pelatihan dapat dilihat pada Gambar 4.



Gambar 4 Perbandingan perambatan *cross-entropy* pada proses pelatihan

Arsitektur AlexNet memerlukan *epoch* relatif lebih sedikit untuk konvergen daripada arsitektur VGG yang ditunjukkan oleh grafik perambatannya yang lebih cepat menurun, meskipun ketiga arsitektur membutuhkan *epoch* yang hampir serupa untuk memenuhi toleransi *cross-entropy* sebesar 0.1, yaitu antara 49 sampai 59 *epoch*. Kecepatan konvergensi ini sebanding dengan banyaknya parameter model CNN dengan arsitektur AlexNet, VGG-11, dan VGG-16 yang digunakan pada penelitian ini yaitu 342619, 484707, dan 490117 parameter. Untuk perbandingan lebih jelas, Tabel 1 menunjukkan jumlah *epoch* dan waktu komputasi pada setiap arsitektur, sedangkan Tabel 2 menunjukkan perbandingan akurasi model pada data *training* dan *testing* menggunakan Persamaan 5.

Tabel 1 Perbandingan *epoch* dan waktu komputasi (detik) setiap arsitektur selama proses pelatihan

No.	**AlexNet**		**VGG-11**		**VGG-16**	
	Epoch	***Waktu***	***Epoch***	***Waktu***	***Epoch***	***Waktu***
1	54	308.91	52	920.97	**54**	6700.03
2	56	303.15	**49**	869.26	57	7537.30
3	**50**	**270.05**	50	**774.59**	59	**6522.80**
Rataan	53	294.04	50	854.94	57	6920.04

Berdasarkan Tabel 1, arsitektur VGG-11 membutuhkan *epoch* yang paling sedikit untuk memenuhi toleransi *cross-entropy* 0.1 dibandingkan dua arsitektur lainnya, dengan *epoch* yang paling sedikit adalah 49 dan yang paling banyak adalah 52. Sementara itu, AlexNet memiliki jumlah *epoch* rata-rata yang lebih sedikit daripada VGG-16. Jumlah *epoch* menunjukkan banyaknya iterasi yang dibutuhkan model untuk memenuhi toleransi yang dibutuhkan. Dari segi waktu komputasi, model yang paling cepat adalah AlexNet dengan rata-rata 294.04 detik atau 4.9 menit. Sementara itu, model VGG-11 membutuhkan waktu rata-rata 854.94 detik atau 14.25 menit dan VGG-16 membutuhkan waktu 6920.04 detik atau hampir dua jam. Waktu komputasi sebanding dengan jumlah parameter yang terkandung dalam model CNN. Artinya, semakin banyak parameter yang terkandung dalam model CNN, maka semakin lama pula waktu komputasi yang diperlukan untuk melatih model. Sebaliknya, waktu komputasi tidak memiliki keterkaitan dengan jumlah *epoch*. Jumlah *epoch* yang sedikit belum tentu butuh waktu komputasi yang lebih sedikit pula, misalnya arsitektur VGG-11. Meskipun memiliki jumlah *epoch* yang relatif lebih sedikit daripada AlexNet, tetapi VGG-11 membutuhkan waktu komputasi yang relatif lebih lama dibandingkan AlexNet. Hal yang sama juga berlaku untuk percobaan pada arsitektur yang sama. Pada model VGG-16, percobaan ketiga memberikan waktu komputasi yang paling sedikit, meskipun jumlah *epoch* yang dibutuhkan lebih banyak dibandingkan yang lainnya.

Tabel 2 Perbandingan akurasi data *training* dan *testing* setiap arsitektur pada proses pelatihan

No.	**AlexNet**		**VGG-11**		**VGG-16**	
	Train	***Test***	***Train***	***Test***	***Train***	***Test***
1	99.12%	97.35%	98.68%	96.46%	**98.46%**	**94.69%**
2	97.58%	96.46%	98.90%	97.35%	98.02%	94.69%
3	**99.34%**	**100.00%**	**99.34%**	**97.35%**	97.14%	94.69%
Rataan	98.68%	97.94%	98.97%	97.05%	97.87%	94.69%

Berdasarkan Tabel 2, ketiga arsitektur memberikan akurasi yang sangat baik dalam mengklasifikasikan ekspresi wajah. Akurasi dari setiap model tidak kurang dari 94% baik untuk data *training* maupun data *testing*. Secara, rata-rata arsitektur AlexNet memberikan akurasi yang paling baik dibandingkan dua arsitektur lainnya, diikuti oleh VGG-11, dan VGG-16. Performa terbaik dari model CNN dengan arsitektur AlexNet terjadi pada percobaan ketiga, dengan akurasi pada data *training* sebesar 99.34% dan akurasi pada data *testing* sebesar 100%. Sementara itu, hasil terbaik untuk VGG-11 adalah pada percobaan ketiga dengan akurasi pada data *training* sebesar 99.34% yang sama dengan AlexNet, tetapi akurasi pada data *testing* lebih kecil yaitu 97.35%. Berdasarkan hasil pada Tabel 2, dapat disimpulkan bahwa semakin kompleks model CNN yang digunakan, tidak menjamin akurasi yang diberikan oleh model tersebutlah yang paling baik. Model VGG mungkin dapat mengklasifikasikan masalah ImageNet dengan 10 ribu kelas dengan lebih baik daripada model AlexNet. Akan tetapi untuk kasus klasifikasi dengan jumlah kelas yang lebih sedikit, arsitektur AlexNet dapat memberikan akurasi yang lebih baik dibandingkan arsitektur VGG.

3.3 Performa Model CNN

Nilai yang dicetak tebal pada Tabel 2 menunjukkan hasil yang terbaik dari masing-masing arsitektur dan digunakan sebagai perwakilan untuk diukur performanya berdasarkan Persamaan 6-11. Gambar 5 menunjukkan *confusion matrix* yang terbentuk dari setiap arsitektur terpilih untuk data *training* dan *testing*.

Confusion Matrix: AlexNet

Data Training

		Prediksi		
		Kelas 1	Kelas 2	Kelas 3
Aktual	Kelas 1	146	3	0
	Kelas 2	0	153	0
	Kelas 3	0	0	152

Data Testing

		Prediksi		
		Kelas 1	Kelas 2	Kelas 3
Aktual	Kelas 1	40	0	0
	Kelas 2	0	36	0
	Kelas 3	0	0	37

Confusion Matrix: VGG -11

Data Training

		Prediksi		
		Kelas 1	Kelas 2	Kelas 3
Aktual	Kelas 1	149	0	0
	Kelas 2	1	151	1
	Kelas 3	1	0	151

Data Testing

		Prediksi		
		Kelas 1	Kelas 2	Kelas 3
Aktual	Kelas 1	39	0	1
	Kelas 2	0	36	0
	Kelas 3	1	1	35

Confusion Matrix: VGG -16

Data Training

		Prediksi		
		Kelas 1	Kelas 2	Kelas 3
Aktual	Kelas 1	148	1	0
	Kelas 2	4	148	1
	Kelas 3	0	1	151

Data Testing

		Prediksi		
		Kelas 1	Kelas 2	Kelas 3
Aktual	Kelas 1	39	1	0
	Kelas 2	1	35	0
	Kelas 3	3	1	33

Gambar 5 Perbandingan confusion matrix untuk setiap arsitektur pada data *training* dan *testing*

Selanjutnya, performa model dihitung menggunakan Persamaan 6-11 untuk data *training* (Tabel 3) dan *testing* (Tabel 4) dari *confusion matrix* yang dihasilkan. *True positive rate* (TPR) atau disebut juga *recall* atau sensitifitas menunjukkan seberapa sering model memprediksi positif, ketika kelas aktualnya positif. Sebaliknya, *positive predictive value* (PPV) atau presisi menunjukkan seberapa sering prediksi itu benar, ketika model memprediksi positif. Sementara itu, F1-*score* merupakan rataan harmonik dari *recall* dan *precision*. Baik untuk data *training* maupun *testing*, performa model CNN dengan arsitektur AlexNet lebih baik dibandingkan arsitektur VGG-11 dan VGG-16. Pada data *training*, F1-*score* untuk arsitektur AlexNet, VGG-11 dan VGG-16 masing-masing adalah 99.3%, 99.3%, dan 98.5%. Sementara pada data *testing*, nilai F1-*score* untuk arsitektur AlexNet, VGG-11 dan VGG-16 masing-masing adalah 100%, 97.4%, dan 94.9%.

Tabel 3 Perbandingan performa model terpilih pada data *training*

Kelas	**AlexNet**			**VGG-11**			**VGG-16**		
	TPR	***PPV***	***F1***	***TPR***	***PPV***	***F1***	***TPR***	***PPV***	***F1***
1	98.0%	100%	99.0%	100%	98.7%	99.3%	99.3%	97.4%	98.3%
2	100%	98.1%	99.0%	98.7%	100%	99.3%	96.7%	98.7%	97.7%
3	100%	100%	100%	99.3%	99.3%	99.3%	99.3%	99.3%	99.3%
Makro	99.3%	99.4%	99.3%	99.3%	99.3%	99.3%	98.5%	98.5%	98.5%

Tabel 4 Perbandingan performa model terpilih pada data *testing*

Kelas	AlexNet			VGG-11			VGG-16		
	TPR	***PPV***	***F1***	***TPR***	***PPV***	***F1***	***TPR***	***PPV***	***F1***
1	100%	100%	100%	97.5%	97.5%	97.5%	97.5%	90.7%	94.0%
2	100%	100%	100%	100%	97.3%	98.6%	97.2%	94.6%	95.9%
3	100%	100%	100%	94.6%	97.2%	95.9%	89.2%	100%	94.3%
Makro	100%	100%	100%	97.4%	97.3%	97.4%	94.6%	95.1%	94.9%

4. KESIMPULAN

Berdasarkan hasil penelitian, dapat disimpulkan bahwa model CNN dengan arsitektur AlexNet, VGG-11, dan VGG-16 dapat mengklasifikasikan tiga ekspresi wajah dari tujuh pengekspresi dengan sangat baik, berdasarkan nilai rata-rata akurasi pada data *training* dan *testing* yang lebih dari 94%. Untuk memenuhi nilai cross-entropy sebesar 0.1, model CNN dengan arsitektur VGG-11 memerlukan jumlah *epoch* paling sedikit dibandingkan arsitektur lainnya. Akan tetapi, arsitektur AlexNet yang memerlukan waktu komputasi paling sedikit. Waktu komputasi sebanding dengan jumlah parameter yang terkandung dalam model CNN. Sebaliknya, waktu komputasi tidak memiliki keterkaitan dengan jumlah *epoch*. Jumlah *epoch* yang sedikit belum tentu membutuhkan waktu komputasi yang lebih sedikit pula. Performa model diperkuat dengan hasil pengukuran yang baik menggunakan recall, presisi, dan F1-score untuk klasifikasi multiclass.

UCAPAN TERIMA KASIH

Peneliti mengucapkan terima kasih kepada Ade Irawan, Evi Ardiyani, Fitra Nuvus Salsabila, dan Revi Refina yang telah bersedia untuk berkontribusi dalam pengumpulan data. Penulis juga mengucapkan terima kasih kepada Departemen Matematika, IPB University atas segala dukungan yang telah diberikan.

DAFTAR PUSTAKA

[1] B. Niu, Z. Gao, and B. Guo, "Facial Expression Recognition with LBP and ORB Features," *Comput. Intell. Neurosci*., 2021, doi: 10.1155/2021/8828245.

[2] A. Mehrabian, "Communication without words," in *Communication Theory: Second Edition*, 2nd ed., D. Mortensen, Ed. New York: Routledge, 2017, pp. 193–200. doi: 10.4324/9781315080918-15.

[3] F. Dornaika and B. Raducanu, "Efficient facial expression recognition for human robot interaction," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2007, vol. 4507 LNCS, pp. 700–708. doi: 10.1007/978-3-540-73007-1_84.

[4] S. Hickson, V. Kwatra, N. Dufour, A. Sud, and I. Essa, "Eyemotion: Classifying facial expressions in VR using eye-tracking cameras," in *Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 1626–1635. doi: 10.1109/WACV.2019.00178.

[5] C. H. Chen, I. J. Lee, and L. Y. Lin, "Augmented reality-based self-facial modeling to promote the emotional expression and social skills of adolescents with autism spectrum disorders," *Res. Dev. Disabil*., vol. 36, pp. 396–403, 2015, doi: 10.1016/j.ridd.2014.10.015.

[6] M. A. Assari and M. Rahmati, "Driver drowsiness detection using face expression recognition," in *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, 2011, pp. 337–341. doi: 10.1109/ICSIPA.2011.6144162.

[7] A. Mahmood, S. Hussain, K. Iqbal, and W. S. Elkilani, "Recognition of Facial Expressions under Varying Conditions Using Dual-Feature Fusion," *Math. Probl. Eng*., vol. 2019, 2019, doi: 10.1155/2019/9185481.

[8] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *Am. Stat.*, vol. 46, no. 3, pp. 175–185, 1992, doi: 10.1080/00031305.1992.10475879.
[9] I. Kotsia and I. Pitas, "Facial expression recognition in image sequences using geometric deformation features and support vector machines," *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 172–187, 2007, doi: 10.1109/TIP.2006.884954.
[10] J. R. Lee, L. Wang, and A. Wong, "EmotionNet Nano: An Efficient Deep Convolutional Neural Network Design for Real-Time Facial Expression Recognition," *Front. Artif. Intell.*, vol. 3, pp. 1–9, 2021, doi: 10.3389/frai.2020.609673.
[11] S. Li and W. Deng, "Deep facial expression recognition: a survey," *J. Image Graph.*, vol. 25, no. 11, pp. 2306–2320, 2020, doi: 10.11834/jig.200233.
[12] W. Mellouk and W. Handouzi, "Facial emotion recognition using deep learning: Review and insights," *Procedia Comput. Sci.*, vol. 175, pp. 689–694, 2020, doi: 10.1016/j.procs.2020.07.101.
[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, vol. 86, no. 11, pp. 2278–2323. doi: 10.1109/5.726791.
[14] M. Z. Alom et al., "A state-of-the-art survey on deep learning theory and architectures," *Electron.*, vol. 8, no. 3, 2019, doi: 10.3390/electronics8030292.
[15] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving Deep Convolutional Neural Networks for Image Classification," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 394–407, 2020, doi: 10.1109/TEVC.2019.2916183.
[16] E. Aydemir and K. Karagul, "Solving a Periodic Capacitated Vehicle Routing Problem Using Simulated Annealing Algorithm for a Manufacturing Company," *Brazilian J. Oper. Prod. Manag.*, vol. 17, no. 1, 2020, doi: 10.14488/bjopm.2020.011.
[17] R. Sells, "Julia Programming Language Benchmark Using a Flight Simulation," *IEEE Aerosp. Conf. Proc.*, 2020, doi: 10.1109/AERO47225.2020.9172277.
[18] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J. Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.
[19] Z. Emersic, D. Stepec, V. Struc, and P. Peer, "Training Convolutional Neural Networks with Limited Training Data for Ear Recognition in the Wild," in *Proceedings - 12th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2017 - 1st International Workshop on Adaptive Shot Learning for Gesture Understanding and Production, ASL4GUP 2017, Biometrics in the Wild, Bwild 2017, Heteroge,* 2017, pp. 987–994. doi: 10.1109/FG.2017.123.
[20] J. Gu et al., "Recent Advances in Convolutional Neural Networks," *Pattern Recognit.*, vol. 77, pp. 354–377, 2018, doi: 10.1016/j.patcog.2017.10.013.
[21] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *COLT 2010 - 23rd Conf. Learn. Theory*, vol. 12, pp. 257–269, 2010.
[22] T. Tieleman and G. E. Hinton, "Coursera: Neural networks for machine learning," 2012.
[23] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, 2015, pp. 1–15.
[24] I. Markoulidakis, G. Kopsiaftis, I. Rallis, and I. Georgoulas, "Multi-Class Confusion Matrix Reduction method and its application on Net Promoter Score classification problem," *ACM Int. Conf. Proceeding Ser.*, vol. 9, no. 4, pp. 412–419, 2021, doi: 10.1145/3453892.3461323.