**Terakreditasi SINTA Peringkat 2**
Surat Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti No. 10/E/KPT/2019
masa berlaku mulai Vol. 1 No. 1 tahun 2017 s.d Vol. 5 No. 3 tahun 2021

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

**JURNAL RESTI**
**(Rekayasa Sistem dan Teknologi Informasi)**
Vol. 5 No. 3 (2021) 489 - 495 ISSN Media Elektronik: 2580-0760

# Analisis Perbandingan Algoritma Klasifikasi MLP dan CNN pada Dataset *American Sign Language*

Mohammad Farid Naufal[1], Sesilia Shania[2], Jessica Millenia[3], Stefan Axel[4], Juan Timothy Soebroto[5], Rizka Febrina P.[6], Mirella Mercifia[7]
[1,2,3,4,5,6,7]Teknik Informatika, Fakultas Teknik, Universitas Surabaya
[1]faridnaufal@staff.ubaya.ac.id, [2]shaniaharsono@gmail.com, [3]jessicamillenia@gmail.com, [4]stefan.07.axel@gmail.com, [5]juan.timothys@gmail.com, [6]riskafebrina29@gmail.com, [7]fiamirella29@gmail.com

***Abstract***

*People who have hearing loss (deafness) or speech impairment (hearing impairment) usually use sign language to communicate. One of the most basic and flexible sign languages is the Alphabet Sign Language to spell out the words you want to pronounce. Sign language uses hand, finger, and face movements to speak the user's thoughts. However, for alphabetical sign language, facial expressions are not used but only gestures or symbols formed using fingers and hands. In fact, there are still many people who don't understand the meaning of sign language. The use of image classification can help people more easily learn and translate sign language. Image classification accuracy is the main problem in this case. This research conducted a comparison of image classification algorithms, namely Convolutional Neural Network (CNN) and Multilayer Perceptron (MLP) to recognize American Sign Language (ASL) except the letters "J" and "Z" because movement is required for both. This is done to see the effect of the convolution and pooling stages on CNN on the resulting accuracy value and F1 Score in the ASL dataset. Based on the comparison, the use of CNN which begins with Gaussian Low Pass Filtering preprocessing gets the best accuracy of 96.93% and F1 Score 96.97%*

*Keywords: CNN, MLP, ASL, classification*

**Abstrak**

Orang yang memiliki gangguan pendengaran (tunarungu) atau gangguan berbicara (tunawicara) biasanya menggunakan bahasa isyarat untuk berkomunikasi. Salah satu bahasa isyarat yang dasar dan fleksibel adalah bahasa isyarat alfabet untuk mengeja kata-kata yang ingin diucapkan. Bahasa isyarat menggunakan gerakan tangan, jari, hingga wajah untuk mengutarakan pikiran penggunanya. Namun, untuk bahasa isyarat alfabet, ekspresi wajah tidak digunakan melainkan hanya gerakan atau simbol yang dibentuk menggunakan jari dan tangan. Nyatanya, masih banyak orang yang tidak mengerti arti dari bahasa isyarat. Penggunaan klasifikasi citra dapat membantu orang untuk lebih mudah mempelajari dan menerjemahkan bahasa isyarat. Akurasi klasifikasi citra menjadi permasalahan utama dalam kasus ini. Penelitian ini melakukan perbandingan algoritma klasifikasi citra yaitu Convolutional Neural Network (CNN) dan Multilayer Perceptron (MLP) untuk mengenali bahasa isyarat alfabet American Sign Language (ASL) kecuali huruf "J" dan "Z" karena diperlukan gerakan untuk keduanya. Hal ini dilakukan untuk melihat efek dari tahapan convolution dan pooling pada CNN terhadap nilai akurasi dan F1 score yang dihasilkan pada dataset ASL. Berdasarkan hasil perbandingan, penggunaan CNN yang diawali dengan tahapan preprocessing Gaussian Low Pass Filter mendapatkan akurasi terbaik yaitu sebesar 96,93% dan F1 Score 96,97%.

Kata kunci: *CNN, MLP, ASL, classification*

## 1. Pendahuluan

Bahasa isyarat (Sign Language) adalah bahasa yang menggunakan idiom tangan dan wajah untuk mengekspresikan pandangan dan pemikiran bagi penyandang cacat (bicara dan mendengar) [1]. Namun nyatanya, tidak banyak orang yang mengetahui atau mempelajari bahasa isyarat. Bahasa ini lebih digunakan oleh sesama tunawicara atau tunarungu. Sehingga komunikasi antara orang tuna wicara atau tuna rungu dengan orang yang dapat mendengar dan berbicara kurang efektif. Pengajaran bahasa isyarat tidak digencarkan di dunia. Karena memang persentase tunarungu di dunia tidaklah besar, yaitu sebanyak 5% atau sekitar 466 juta manusia pada tahun 2018 [2]. Diperkirakan, di tahun 2050, penderita gangguan

Diterima Redaksi: 10-03-2021 | Selesai Revisi: 24-04-2021 | Diterbitkan Online: 20-06-2021

pendengaran akan mencapai angka 900 juta [3]. Komunikasi dengan orang tunarungu atau tunawicara akan jauh lebih dipermudah dengan penggunaan bahasa isyarat. Salah satu bahasa isyarat dasar yang bersifat fleksibel adalah bahasa isyarat untuk alfabet [4]. Tapi sebenarnya, masih lebih luas lagi cakupan bahasa isyarat, termasuk banyaknya jenis bahasa isyarat berbeda yang digunakan di tiap negaranya.

American Sign Language (ASL) adalah salah satu bahasa isyarat yang paling banyak digunakan di dunia dan bahasa keempat yang paling banyak digunakan di Amerika Utara [1]. ASL juga digunakan di Kanada, Meksiko, Afrika Barat, dan Asia. Lebih dari 20 negara lain seperti Jamaika, Panama, Thailand, Malaysia di mana bahasa Inggris adalah bahasa komunikasi utama yang menggunakan ASL untuk komunikasi komunitas mereka yang sulit mendengar.

Penggunaan image classification dan recognition [6] untuk mengetahui bahasa isyarat alfabet dasar dapat mempermudah dan membuat proses belajar bahasa isyarat lebih efektif. Akurasi dari proses klasifikasi citra pada dataset ASL menjadi hal yang penting. Terdapat banyak algoritma klasifikasi citra seperti K Nearest Neighbors (KNN), Support Vector Machine (SVM) [7], Multilayer Perceptron (MLP) [8], dan Convolutional Neural Network (CNN) [9].

Ameen et al. [10] melakukan klasifikasi ASL menggunakan ConvNet dengan memanfaatkan fitur image intensity dan depth data. Namun penelitian ini tidak menambahkan tahapan preprocessing. Tolentino et al. [11] menggunakan CNN dengan jumlah filter 16, kernel 2x2, dan max pooling 2x2 untuk klasifikasi ASL. Penelitian ini menghasilkan accuracy sebesar 93.67%. Dikarenakan menggunakan arsitektur CNN dengan model convolution yang sederhana, ada peluang untuk meningkatkan akurasinya dengan cara membuat arsitektur yang lebih kompleks. Daroya et al. [12] menggunakan Densely Connected Convolutional Neural Network (DenseNet) dan memperoleh akurasi 90.3%. Namun penelitian ini tidak menyebutkan waktu training yang dibutuhkan untuk membentuk model.

MLP dan CNN merupakan bagian dari algoritma Artificial Neural Network (ANN) yang memiliki performa baik untuk klasifikasi citra [13]. MLP disebut juga sebagai vanilla ANN yaitu Neural Network tanpa adanya tahapan pengenalan fitur citra [14], sedangkan CNN memiliki tahapan pengenalan fitur yaitu convolution dan max pooling [15].

Tujuan penelitian ini pertama adalah melakukan perbandingan algoritma MLP dan CNN untuk klasifikasi dataset ASL. Penelitian ini melihat seberapa berpengaruh tahapan pengenalan fitur (convolution dan max pooling) pada CNN terhadap akurasi yang dihasilkan pada dataset ASL jika dibandingkan dengan MLP. Kedua, penelitian ini menambahkan tahapan preprocessing sebelum dataset ASL diklasifikasikan. Penelitian ini melihat seberapa berpengaruh tahapan preprocessing yaitu Gaussian Low Pass Filter dan Laplacian High Pass Filter terhadap akurasi yang dihasilkan oleh setiap algoritma klasifikasi. Ketiga, penelitian ini melihat berapa waktu yang dibutuhkan untuk membentuk model klasifikasi. Pada tahapan uji coba, penelitian ini menggunakan 2 buah arsitektur CNN untuk dibandingkan performanya.

## 2. Metodologi Penelitian

Metode yang digunakan dalam penelitian ini terdiri dari 5 tahap. Gambar 2 menunjukkan alur penelitian ini. Metode penelitian yang dilakukan dimulai dari pengumpulan dataset, preprocessing menggunakan Gaussian LPF dan Laplacian HPF, pemilihan model algoritma, uji coba, dan perbandingan performa.

### 2.1. Pengumpulan Dataset

*Dataset* yang digunakan adalah dataset Sign Language MNIST yang diambil dari kaggle.com [16]. Dataset ini berisi Bahasa isyarat Amerika (ASL) yang berupa gambar tangan yang mewakili 24 kelas huruf dari A-Z kecuali huruf "J" dan "Z". Hal ini dikarenakan huruf "J" dan "Z" tidak dapat digambarkan dengan sebuah gambar tangan saja, karena memerlukan gerakan tangan. Gambar 1 menunjukkan contoh dataset ASL yang diperoleh dari Kaggle [16]. Setiap gambar dalam dataset berukuran 28x28 pixel dengan nilai grayscale antara 0-255. Dataset ini terdiri dari 24 kelas data dengan *train data* sebanyak 17.455 gambar dan test data sebanyak 7.172 gambar. Porsi dataset training dan testing mengikuti aturan yang sudah ada di kaggle yang bertujuan untuk kemungkinan perbandingan dengan penelitian lain yang menggunakan dataset yang sama.

Gambar 1. Contoh Dataset ASL

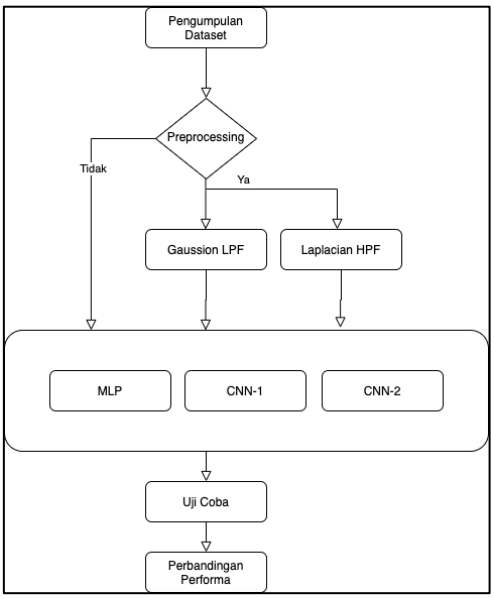
### 2.2. Preprocessing

Metode preprocessing yang digunakan adalah Gaussian Low Pass Filtering dan Laplacian High Pass Filtering karena dapat menghilangkan noise dan mempertajam citra [17]. Tahapan preprocessing dilakukan setelah *dataset* dibaca dan dimasukkan ke dalam variabel.

Gaussian Low Pass Filtering merupakan salah satu bentuk filtering dalam frequency domain yang digunakan untuk menghaluskan gambar. Hal ini dapat

DOI: https://doi.org/10.29207/resti.v5i3.3009

membantu mengurangi noise (titik-titik pada gambar yang menurunkan kualitas gambar) sehingga diharapkan dapat mempermudah algoritma klasifikasi dalam proses pengenalan gambar. Cara kerja filter ini adalah mengubah input gambar yang sudah diubah ke bentuk frekuensi, mengkalikannya dengan mask, lalu mengembalikannya menjadi bentuk spasial, yaitu gambar yang sudah diperhalus. Gambar 3 menunjukkan hasil preprocessing menggunakan Gaussian Low Pass Filtering.




Gambar 2. Metodologi Penelitian

Gambar 3. Hasil Preprocessing Gaussian Low Pass Filter

Fungsi untuk membuat *mask* milik *Gaussian Low Pass Filtering* adalah seperti pada persamaan 1, dengan $H(u,v)$ adalah nilai mask yang dihasilkan, $D(u,v)$ adalah jarak dari titik tengah gambar dalam domain frekuensi ke setiap pixel $(u,v)$ pada gambar, $\sigma$ adalah jarak dari titik tengah gambar hingga frekuensi tertentu.

$$H(u,v) = e^{-D^2(u,v)/2\sigma^2} \quad (1)$$

Laplacian Filtering merupakan salah satu bentuk filtering dalam spatial domain yang digunakan untuk mempertajam gambar. Laplacian merupakan operator linear. Gambar 4 menunjukkan hasil preprocessing menggunakan Laplacian Filtering. Untuk membuat perhitungan menjadi bentuk diskrit, kita menggunakan rumus seperti pada persamaan 2:

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \quad (2)$$

Gambar 4. Hasil Preprocessing Laplacian Filtering

Pada penelitian ini MLP dan CNN terbagi menjadi 3 versi yaitu tanpa preprocessing, dengan Gaussian Low Pass Filtering, dan dengan Laplacian Filtering. Hal ini dilakukan untuk melihat seberapa berpengaruh tahapan dan jenis preprocessing terhadap performa akurasi yang dihasilkan.

### 2.3. Pemilihan Model Algoritma

Implementasi Multilayer Perceptron (MLP) membutuhkan parameter hidden layer, activation, solver, dan jumlah maksimal iterasi. Tabel 1 menunjukkan parameter dan model MLP yang digunakan. Sedangkan Gambar 5 menunjukkan arsitektur MLP. MLP yang digunakan terdiri dari 3 layer, yaitu input layer yang terdiri dari 784 (28x28) input sesuai dengan jumlah pixel citra input, 200 neuron di hidden layer, dan 25 neuron di output layer sesuai dengan jumlah class klasifikasi ASL.

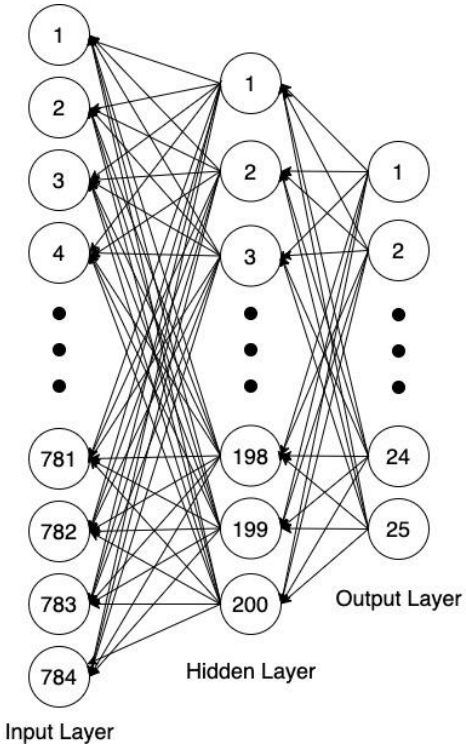
Secara default library Scikit-Learn [18] memiliki parameter hidden layer sebanyak 100 neuron, namun dengan pertimbangan untuk meningkatkan akurasi, maka penelitian ini menggunakan 200 neuron pada hidden layer. Untuk fungsi aktivasi, dapat dikatakan bahwa multilayer tidak akan mengalami kenaikan daya komputasi jika menggunakan fungsi aktivasi linear. Maka dari itu, MLP memakai fungsi aktivasi non-linear, hampir semua fungsi non-linear dapat digunakan pada MLP. Saat ini, fungsi aktivasi non-linear yang popular dipakai adalah logistic/sigmoid [14]. Sedangkan untuk solver atau optimizer penelitian ini menggunakan Limited-memory BFGS (LBFGS) dengan pertimbangan

DOI: https://doi.org/10.29207/resti.v5i3.3009

kecepatan waktu komputasi untuk dataset yang fiturnya tidak kompleks [19].

Tabel 1. Parameter MLP

| Parameter | Value |
|---|---|
| Hidden layer sizes | (200,) |
| Activation | logistic |
| Solver | lbfgs |
| Max. Iteration | 300 |

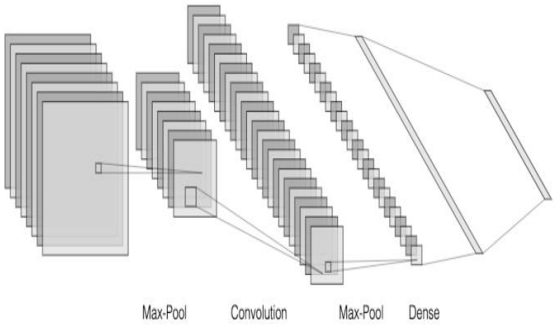



Gambar 5. Hasil Preprocessing Laplacian Filtering

Implementasi Convolutional Neural Network (CNN) menggunakan library keras [20]. Secara umum Gambar 6 menunjukkan arsitektur dari CNN. Penerapan algoritma CNN dibuat 2 versi model sebagai perbandingan untuk dianalisis tingkat akurasinya, yaitu:

1. CNN-1 yang terdiri dari 3 proses convolution, 2 proses pooling, dan 3 fully connected layer. Detail model CNN-1 dapat dilihat di Tabel 2.
2. CNN-2 yang terdiri dari 4 proses convolution, 2 proses pooling, dan 3 fully connected layer. Detail model CNN-1 dapat dilihat di Tabel 3.

Parameter yang dibutuhkan untuk pembentukan model CNN adalah activation function, kernel size, tipe pooling, dan dropout. Kedua model CNN menggunakan parameter dropout untuk menghindari terjadinya overfitting. Sedangkan Dense pada hidden layer menggunakan activation function ReLu dan Dense pada output layer menggunakan Softmax.




Gambar 6. Arsitektur Umum CNN

Tabel 2. Parameter Model CNN-1

| No | Layer | Output Shape | Deskripsi |
|---|---|---|---|
| 1 | Conv2D | (26, 26, 128) | 3x3, ReLu |
| 2 | MaxPooling2D | (13, 13, 128) | 2x2 |
| 3 | Conv2D | (11, 11, 128) | 3x3, ReLu |
| 4 | MaxPooling2D | (5, 5, 128) | 2x2 |
| 5 | Dropout | (5, 5, 128) | 20% |
| 6 | Flatten | (3200) | - |
| 7 | Dropout | (3200) | 20% |
| 8 | Dense | (256) | ReLu |
| 9 | Dropout | (256) | 40% |
| 10 | Dense | (128) | ReLu |
| 11 | Dropout | (128) | 40% |
| 12 | Dense | (25) | Softmax |

Tabel 3. Parameter Model CNN-2

| No | Layer | Output Shape | Deskripsi |
|---|---|---|---|
| 1 | Conv2D | (28, 28, 32) | 3x3, ReLu |
| 2 | Dropout | (28, 28, 32) | 20% |
| 4 | Conv2D | (28, 28, 64) | 3x3, ReLu |
| 5 | MaxPooling2D | (14, 14, 64) | 2x2 |
| 6 | Dropout | (14, 14, 64) | 20% |
| 8 | Conv2D | (14, 14, 64) | 3x3, ReLu |
| 9 | MaxPooling2D | (7, 7, 64) | 2x2 |
| 10 | Dropout | (7, 7, 64) | 20% |
| 12 | Conv2D | (7, 7, 128) | 3x3, ReLu |
| 13 | Dropout | (7, 7, 128) | 20% |
| 15 | Flatten | (6272) | - |
| 16 | Dropout | (6272) | 20% |
| 17 | Dense | (256) | ReLu |
| 18 | Dropout | (256) | 20% |
| 19 | Dense | (128) | ReLu |
| 20 | Dropout | (128) | 20% |
| 21 | Dense | (25) | Softmax |

### 2.4. Tahapan Uji Coba

Tahapan uji coba dilakukan dengan melakukan proses training menggunakan MLP, CNN-1, dan CNN-2. Selanjutnya dilakukan proses testing dengan menghitung accuracy dan F1 Score. Persamaan 3 menunjukkan rumus perhitungan accuracy, dan Persamaan 4 menunjukkan rumus perhitungan F1 Score.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

$$F1\ Score = 2 * \frac{Precision\ x\ Recall}{Precision+Recall} \quad (4)$$

DOI: https://doi.org/10.29207/resti.v5i3.3009

Tabel 4 menunjukkan daftar algoritma yang dikombinasikan dan tanpa dikombinasikan dengan tahapan preprocessing yang digunakan pada penelitian ini.

Tabel 4. Daftar Algortima dan Preprocessing

| No | Algoritma |
|---|---|
| 1 | MLP |
| 2 | MLP + Gaussian LPF |
| 4 | MLP + Laplacian HPF |
| 5 | CNN-1 |
| 6 | CNN-1 + Gaussian LPF |
| 8 | CNN-1 + Laplacian HPF |
| 9 | CNN-2 |
| 10 | CNN-2 + Gaussian LPF |
| 12 | CNN-2 + Laplacian HPF |

Proses training pada MLP menggunakan Iterasi maksimal sebesar 300 sedangkan pada CNN-1 dan CNN-2 menggunakan parameter batch size sebesar 64 dan 25 epoch. Resource untuk menjalankan proses training dan testing adalah server yang disediakan oleh Google Colab [21]. Tabel 5 menunjukkan spesifikasi environment Goole Colab.

Tabel 5. Spesifikasi Resource Google Colab

| No | Parameter | Spesifikasi |
|---|---|---|
| 1 | CPU Model Name | Intel® Xeon® |
| 2 | CPU Frequencies | 2.30 GHz |
| 4 | CPU Cores | 2 |
| 5 | RAM | 12 GB |
| 6 | Space of Disk | 25 GB |
| 8 | GPU Mode Name | Nvidia K80 |
| 9 | GPU Memory | 12 GB |

### 2.5. Perbandingan Performa

Pada tahapan ini dilakukan perbandingan dari performa accuracy dan F1 Score dari setiap algoritma dengan dan tanpa preprocessing. Selain accuracy dan F1 Score, waktu training juga dibandingkan. Hal ini dilakukan untuk mengetahui algoritma mana yang paling cocok dan efisien pada dataset ASL.

## 3. Hasil dan Pembahasan

Dari hasil penelitian yang telah dilakukan performa mengenai accuracy dan F1 Score dapat dilihat pada Tabel 6. Terlihat jelas bahwa MLP tanpa preprocessing mendapatkan hasil akurasi yang lebih baik dibandingkan MLP yang menggunakan preprocessing baik itu Laplacian HPF maupun Gaussian LPF. Dapat disimpulkan bahwa akurasi terbaik dari MLP sebesar 74,79% dan tahapan preprocessing Gaussian LPF tidak berpengaruh signifikan terhadap peningkatan performa pada algoritma MLP. Selain itu tahapan preprocessing Laplacian HPF menurunkan performa MLP. Sebagai contoh untuk model CNN-2 + Gaussian LPF perhitungan accuracy dan F1 Score dengan menggunakan persamaan 3 dan 4 sebagai berikut.

$$Accuracy = \frac{6952}{6952 + 220} = 96.93\%$$

$$F1\ Score = 2 * \frac{96.93\ x\ 96.93}{96.93 + 96.93} = 96.93\%$$

Tabel 6. Daftar Algortima dan Preprocessing

| No | Algoritma | Accuracy | F1 Score |
|---|---|---|---|
| 1 | MLP | 74,79 | 74,79 |
| 2 | MLP + Gaussian LPF | 74,79 | 74,76 |
| 4 | MLP + Laplacian HPF | 30,34 | 30,34 |
| 5 | CNN-1 | 96,39 | 96,39 |
| 6 | CNN-1 + Gaussian LPF | 95,41 | 95,41 |
| 8 | CNN-1 + Laplacian HPF | 95,05 | 95,05 |
| 9 | CNN-2 | 96,10 | 96,10 |
| 10 | CNN-2 + Gaussian LPF | 96,93 | 96,93 |
| 12 | CNN-2 + Laplacian HPF | 96,82 | 96,82 |

CNN-2 yang ditambahkan dengan preprocessing Gaussian LPF mendapatkan nilai accuracy dan F1 Score terbaik. Gambar 7 menunjukkan confusion matrix dari CNN-2 + Gaussian LPF.

```
[[331   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0 432   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0 310   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0 245   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 497   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0 247   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0 328   0   0   0   0   0   0   0   0   0   1   0  19   0   0   0   0   0   0]
 [  0   0   0   0   0   0  16 420   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 288   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0 270   0   0   0   0   0   0  40   0   0  21   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 209   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  21   0   0   0   0   0   0 354   0   0   0   0   0  19   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  22 269   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 246   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 347   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 164   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 144   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 246   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 227   0   0   0  21   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 266   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 346   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 206   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 267   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  39   0   0   0   0   0   0 293]]
```

Gambar 7. Confusion Matrix CNN-2 + Gaussian LPF

Dapat disimpulkan bahwa model CNN-2 yang memiliki arsitektur lebih kompleks dibandingkan CNN-1 dan ditambahkan preprocessing Gaussian HPF berdampak terhadap peningkatan performa CNN. Selain itu algoritma CNN jauh lebih unggul daripada MLP. Tahapan Convolution dan Max Pooling pada CNN terbukti meningkatkan performa sangat signifikan.

Tabel 7 menunjukkan perbandingan waktu training dari tiap algoritma. Waktu didapatkan dengan cara menghitung lama waktu proses training dengan spesifikasi komputer seperti yang ada di Tabel 5. Algoritma CNN membutuhkan waktu training yang lebih lama dibandingkan MLP karena terdapat lebih banyak proses yang dilakukan. CNN-2 membutuhkan waktu dalam pembentukan model lebih lama dibandingkan CNN-1, karena arsitektur yang dimiliki lebih kompleks. Penggunaan preprocessing juga berpengaruh terhadap waktu training baik untuk MLP dan CNN. Perbedaan waktu pembentukan model lebih terlihat di CNN-2 yaitu penggunaan Gaussian LPF membutuhkan waktu lebih lama dibandingkan Laplacian HPF. Namun pada CNN-1 dan MLP waktu

DOI: https://doi.org/10.29207/resti.v5i3.3009

preprocessing tidak berbeda antara Gaussian LPF dan Laplacian HPF.

Tabel 7. Waktu Training Tiap Algoritma

| No | Algoritma | Waktu Training (Menit) |
|---|---|---|
| 1 | MLP | 10 |
| 2 | MLP + Gaussian LPF | 15 |
| 4 | MLP + Laplacian HPF | 15 |
| 5 | CNN-1 | 34 |
| 6 | CNN-1 + Gaussian LPF | 33 |
| 8 | CNN-1 + Laplacian HPF | 33 |
| 9 | CNN-2 | 55 |
| 10 | CNN-2 + Gaussian LPF | 69 |
| 12 | CNN-2 + Laplacian HPF | 54 |

## 4. Kesimpulan

Berdasarkan uji coba yang dilakukan dapat disimpulkan bahwa akurasi yang didapatkan dalam penelitian ini cukup baik. Algoritma yang memiliki performa terbaik adalah CNN-2 yang disertai dengan preprocessing menggunakan Gaussian LPF. Algortima ini menghasilkan akurasi sebesar 96,97% dan F1 Score 96.93%. Sedangkan pada algoritma MLP didapatkan akurasi terbaik sebesar 74,79% tanpa disertai preprocessing. Pada kasus ini juga algoritma CNN cenderung menghasilkan nilai akurasi yang lebih baik dibandingkan Multilayer Perceptron (MLP). Semakin banyak layer pada model CNN, semakin baik pula akurasi yang didapatkan. Penambahan tahapan preprocessing Gaussian LPF dapat meningkatkan akurasi dan F1 Score pada CNN jika dibandingkan dengan Laplacian HPF di semua model MLP dan CNN.

Semakin banyak layer menyebabkan waktu training atau pembentukan model menjadi semakin lama. Hal ini terbukti rata-rata waktu training dari CNN adalah 3 hingga 4 kali lebih lama jika dibandingkan MLP.

Berdasarkan hasil temuan pada penelitian ini, maka saran untuk penelitian selanjutnya adalah:

1. Penggunaan algoritma CNN dengan model yang lebih bervariasi lagi, seperti LeNet-5 [22], AlexNet [23], ZFNet [24], GoogLeNet [25], VGGNet [26], ResNet [27].
2. Implementasi metode *preprocessing* yang lebih bervariasi, seperti misalnya Butterworth High Pass Filtering atau Gaussian High Pass Filtering [17].
3. Mengimplementasikan sebuah aplikasi real time untuk menerjemahkan ASL.

## Daftar Rujukan

[1] Shivashankara and Srinath, "American Sign Language Recognition System: An Optimal Approach," *Int. J. Image, Graph. Signal Process.*, vol. 10, no. 8, pp. 18–30, 2018, doi: 10.5815/ijigsp.2018.08.03.

[2] "Deafness." https://www.who.int/news-room/facts-in-pictures/detail/deafness (accessed Dec. 16, 2020).

[3] WHO, "Deafness and hearing loss." https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss (accessed Dec. 16, 2020).

[4] W. C. Stokoe and M. Marschark, "Sign language structure: An outline of the visual communication systems of the american deaf," *J. Deaf Stud. Deaf Educ.*, vol. 10, no. 1, pp. 3–37, 2005, doi: 10.1093/deafed/eni001.

[5] N. I. on D. and O. Communication, "What Is American Sign Language (ASL)? | NIDCD." https://www.nidcd.nih.gov/health/american-sign-language (accessed Dec. 16, 2020).

[6] Y. Baştanlar and M. Ozuysal, *Introduction to Machine Learning Second Edition*, vol. 1107. 2014.

[7] J. Kim, B.-S. Kim, and S. Savarese, "Comparing Image Classification Methods: K-Nearest-Neighbor and Support-Vector-Machines," *Appl. Math. Electr. Comput. Eng.*, pp. 133–138, 2012.

[8] N. Coskun and T. Yildirim, "The effects of training algorithms in MLP network on image classification," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2, no. see 17, pp. 1223–1226, 2003, doi: 10.1109/ijcnn.2003.1223867.

[9] M. Xin and Y. Wang, "Research on image classification model based on deep convolution neural network," *Eurasip J. Image Video Process.*, vol. 2019, no. 1, 2019, doi: 10.1186/s13640-019-0417-8.

[10] S. Ameen and S. Vadera, "A convolutional neural network to classify American Sign Language fingerspelling from depth and colour images," *Expert Syst.*, vol. 34, no. 3, 2017, doi: 10.1111/exsy.12197.

[11] L. K. S. Tolentino, R. O. S. Juan, A. C. Thio-ac, M. A. B. Pamahoy, R. R. Forteza, and X. J. O. Garcia, "Static Sign Language Recognition Using Deep Learning," no. November, 2019, doi: 10.18178/ijmlc.2019.9.6.879.

[12] R. Daroya, D. Peralta, and P. Naval, "Alphabet Sign Language Image Classification Using Deep Learning," *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2018-October, no. October, pp. 646–650, 2019, doi: 10.1109/TENCON.2018.8650241.

[13] S. Ben Driss, M. Soua, R. Kachouri, and M. Akil, "A comparison study between MLP and convolutional neural network models for character recognition," *Real-Time Image Video Process. 2017*, vol. 10223, p. 1022306, 2017, doi: 10.1117/12.2262589.

[14] P. Marius-Constantin, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Trans. Circuits Syst.*, vol. 8, no. 7, pp. 579–588, 2009.

[15] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," pp. 1–11, 2015, [Online]. Available: http://arxiv.org/abs/1511.08458.

[16] tecperson, "Sign Language MNIST | Kaggle," 2017. https://www.kaggle.com/datamunge/sign-language-mnist (accessed Dec. 18, 2020).

[17] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, 4e*. .

[18] "scikit-learn: machine learning in Python — scikit-learn 0.23.2 documentation." https://scikit-learn.org/stable/ (accessed Dec. 18, 2020).

[19] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program. Ser. B*, vol. 45, no. 3, pp. 503–528, 1989.

[20] F. Chollet and & O., "Keras: the Python deep learning API," *Keras: the Python deep learning API*, 2020. https://keras.io/ (accessed Dec. 18, 2020).

[21] Google Colab, "Welcome to Colaboratory - Colaboratory," *Getting Started - Introduction*, 2020. https://colab.research.google.com/notebooks/intro.ipynb (accessed Dec. 18, 2020).

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Ha, "LeNet," *Proc. IEEE*, no. November, pp. 1–46, 1998.

[23] T. F. Gonzalez, "Handbook of approximation algorithms and metaheuristics," *Handb. Approx. Algorithms Metaheuristics*, pp. 1–1432, 2007, doi: 10.1201/9781420010749.

DOI: https://doi.org/10.29207/resti.v5i3.3009

[24] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014, doi: 10.1007/978-3-319-10590-1_53.

[25] C. Szegedy *et al.*, "Going deeper with convolutions," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 1–9, 2015, doi: 10.1109/CVPR.2015.7298594.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

DOI: https://doi.org/10.29207/resti.v5i3.3009