Model Penerjemah Bahasa Isyarat SIBI Statis Berbasis Convolutional Neural Network

Ni Made Wipra Ranum Ratnayu[a1], I Dewa Made Bayu Atmaja Darmawan[a2],
I Putu Gede Hendra Suputra[a3]

[a]Program Studi Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam,
Universitas Udayana
Jalan Raya Kampus Udayana, Bukit Jimbaran, Kuta Selatan, Badung, Bali, Indonesia
[1]ratnayu.2208561042@student.unud.ac.id
[2]dewabayu@unud.ac.id
[3]dhendra.suputra@unud.ac.id

Abstract

This study addresses the communication gap between deaf and hearing communities by developing an optimal sign language recognition system for Indonesian Sign Language System (SIBI) static gestures. A comprehensive comparative analysis was conducted on four VGG architecture variants (VGG-11, VGG-13, VGG-16, and VGG-19) using a dataset across 10 SIBI word classes. The research employed systematic methodology including data extraction from video sources, preprocessing with augmentation techniques, model training over 25 epochs, and comprehensive evaluation using accuracy, precision, recall, and F1-score metrics. Results demonstrate that VGG-16 achieves superior performance with 83.4% accuracy, 85.2% precision, 83.4% recall, and 82.9% F1-score, establishing optimal balance between model complexity and generalization capability. The study reveals diminishing returns phenomenon in VGG-19 despite increased architectural complexity. Computational efficiency analysis shows VGG-11 provides highest efficiency score (10.46 GFLOPs) while VGG-16 maintains optimal accuracy-efficiency trade-off. These findings provide crucial insights for developing effective assistive technology solutions that bridge communication barriers for the Indonesian deaf community.

Keywords: *sign language recognition, convolutional neural network, VGG architecture, SIBI classification, computer vision, assistive technology*

1. Pendahuluan

Komunikasi merupakan kebutuhan dasar manusia dalam berinteraksi dan menyampaikan informasi[1], [2], [3]. Bagi penyandang tunarungu dan tunawicara, komunikasi dilakukan melalui bahasa isyarat[4] yang merupakan bahasa *visual gestural* menggunakan gerakan tangan, ekspresi wajah, dan postur tubuh[5]. Di Indonesia, Sistem Isyarat Bahasa Indonesia (SIBI) telah ditetapkan sebagai salah satu bahasa isyarat resmi yang digunakan dalam komunikasi formal[6], khususnya dalam lingkungan pendidikan dan pelayanan publik. Namun dalam praktiknya, bagi penyandang tunarungu komunikasi perlu difasilitasi dengan bahasa khusus yang sesuai untuk kebutuhan sehari-hari agar dapat berkomunikasi dan memahami komunikasi[7]. Namun masalah utama yang dihadapi adalah adanya kesenjangan komunikasi antara penyandang tunarungu dengan masyarakat umum yang tidak memahami bahasa isyarat, sehingga diperlukan solusi teknologi yang dapat menjembatani kesenjangan komunikasi ini[8].

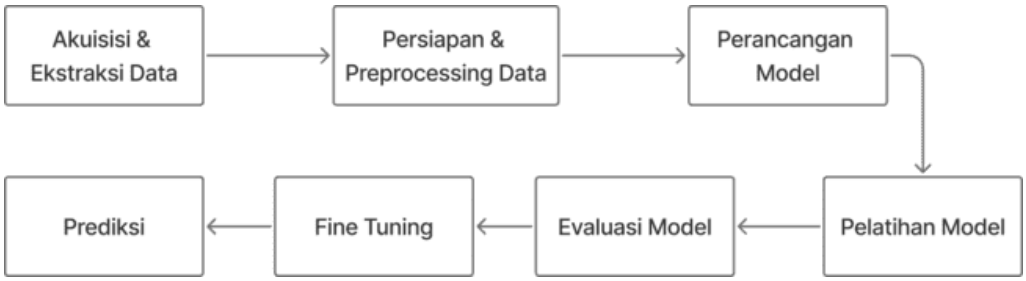
Perkembangan teknologi kecerdasan buatan seperti pada *computer vision* telah membuka peluang besar untuk pengembangan bahasa isyarat[9]. Peneliti sebelumnya sudah pernah melakukan penelitian tentang Bahasa Isyarat Indonesia (BISINDO) statis menggunakan *Convolutional Neural Networ*k (CNN) telah terbukti sebagai metode yang efektif untuk klasifikasi citra, termasuk dalam pengenalan bahasa isyarat[3]. Seperti yang dikemukakan oleh Sholawati dkk, "*Convolutional Neural Network* merupakan salah satu jenis algoritma *neural network* yang didesain untuk memproses data citra"[6].

Implementasi arsitektur CNN sebagai pengenalan bahasa isyarat SIBI telah menunjukkan hasil yang menjanjikan dalam beberapa penelitian terdahulu. Rivan & Hartoyo berhasil mengembangkan sistem klasifikasi isyarat bahasa Indonesia menggunakan arsitektur VGG-16 dan AlexNet dengan tingkat akurasi yang tertinggi mencapai 99,32% pada setiap huruf menggunakan VGG-16 dengan optimizer Adam[10]. Demikian pula, penelitian oleh Sholawati dkk yang mengembangkan aplikasi pengenalan bahasa isyarat abjad SIBI berbasis web menggunakan CNN mencapai akurasi sebesar 80,76%[6]. Siddik juga melakukan perbandingan antara VGG-16 dan ResNet-50 untuk klasifikasi *hand sign language digits* dan menemukan bahwa VGG-16 memberikan hasil terbaik dengan akurasi sebesar 97,29%, presisi sebesar 97,38%, recall sebesar 97,45%, dan F1 score sebesar 97,36%[11]. Meski menunjukkan hasil positif, penelitian-penelitian sebelumnya memiliki beberapa keterbatasan. Sebagian besar penelitian berfokus pada pengenalan gestur dinamis atau *real-time* yang memerlukan komputasi yang besar, sedangkan pengenalan bahasa isyarat statis masih memerlukan eksplorasi lebih lanjut. Sebagaimana dikemukakan oleh Giamiko & Tjiong, pemilihan arsitektur yang tepat dan dataset yang sesuai dapat membantu menghasilkan model klasifikasi yang lebih akurat[12].

Pada penelitian sebelumnya menyatakan bahwa arsitektur VGG mendapat hasil akurasi yang optimal untuk menangatasi masalah klasifikasi maupun deteksi[13], [14]. Secara lebih spesifik, VGG-16 mengungguli varian lain dari segi hasil yang diharapkan dibanding arsitektur VGG lainnya[11], [15], [16]. Untuk memastikan hal tersebut, peneliti ingin membandingkan berbagai varian arsitektur VGG khususnya dalam pengenalan bahasa isyarat SIBI statis. Penelitian ini bertujuan untuk mengembangkan model terbaik penerjemah bahasa isyarat SIBI statis dengan membandingkan performa berbagai arsitektur VGG (VGG-11, VGG-13, VGG-16, dan VGG-19). Melalui evaluasi komprehensif terhadap keempat varian arsitektur tersebut, penelitian ini diharapkan dapat memberikan kontribusi dalam menentukan arsitektur VGG yang paling optimal untuk klasifikasi bahasa isyarat SIBI statis dan mengetahui beban komputasi dari setiap arsitektur menggunakan *Giga Floating Point Operations* (GFLOPs), sehingga dapat mendukung pengembangan teknologi *assistive* yang lebih baik bagi komunitas tunarungu di Indonesia.

2. Metode Penelitian

Penelitian ini mengadopsi pendekatan eksperimental dengan tahapan terstruktur yang meliputi akuisisi dan ekstraksi data, persiapan dan *preprocessing* data, perancangan model, pelatihan model, evaluasi model, *fine tuning*, hingga implementasi sistem dengan prediksi. Setiap tahap dirancang secara sistematis untuk memastikan validitas data dan reliabilitas model dalam melakukan klasifikasi isyarat statis SIBI. Diagram alur keseluruhan proses dapat dilihat pada Gambar 1 yang menggambarkan seluruh alur proses penelitian.



Gambar 1. Alur Proses Penelitian

2.1 Akuisisi dan Ekstraksi Data

Dataset yang digunakan diperoleh dari *Indonesian Sign Language System* (SIBI) Dataset: *Sentences Enhanced by Diverse Facial Expressions for Total Communication[17]*. Dataset tersebut terdiri atas video beresolusi tinggi (1280×720 piksel) dengan 30 *frame* per detik, yang mencakup kalimat beranotasi dalam SIBI serta tujuh jenis ekspresi wajah berbeda. Dari kumpulan video tersebut, dipilih 10 video yang berisi kata sederhana, yaitu: ajak, aku, bapak, buku, dia, ibu, kakak, kamu, saya, dan tugas. Kata-kata yang dipilih merupakan kata yang paling sering

p-ISSN: 2986-3929
e-ISSN: 3032-1948

digunakan dan umum serta banyak digunakan. Dari kumpulan video kata tersebut kemudian dibagi menjadi beberapa *frame* sesuai panjang video lalu diambil 6 *frame* yang memiliki perbedaan visual. Kemudian *frame* yang dipilih akan disimpan sebagai dataset statis. Total *frame* yang dijadikan image untuk dataset statis berjumlah 4250 data untuk pelatihan, 350 data untuk validasi, dan 350 data untuk testing. Masing-masing 425 data perkata untuk data latih, 35 data per kata untuk validasi, dan 35 data perkata untuk testing.

Ekstraksi video menjadi kumpulan citra statis dilakukan melalui proses *key-frame selection* berbasis perhitungan perbedaan visual antar *frame*. Metode *structural similarity index* (SSIM) dan *color histogram difference* digunakan untuk memilih *frame-frame* yang menunjukkan perubahan posisi tangan atau ekspresi wajah yang signifikan. Seleksi ini bertujuan untuk mengurangi redundansi visual dan memastikan data yang digunakan merepresentasikan variasi gerakan secara optimal seperti pada Gambar 2.

Gambar 2. Potongan Video Menjadi *Image* dengan *Frame*

2.2 Persiapan Data dan *Preprocessing* Data

Setelah diperoleh kumpulan citra statis, dilakukan tahap praproses yang dimulai dengan pengaturan struktur direktori data menggunakan 10 kata yang sudah dipilih. Setiap kelas kata dibuat dalam direktori terpisah untuk memudahkan proses loading dan pelabelan otomatis. Struktur organisasi data ini memungkinkan sistem untuk mengidentifikasi dan memuat gambar berdasarkan kelas yang sesuai dengan gestur bahasa isyarat yang direpresentasikan.

Seluruh citra dinormalisasi ke resolusi 224×224 piksel agar sesuai dengan arsitektur model VGG yang akan digunakan. Untuk meningkatkan variabilitas dataset dan mencegah *overfitting*, diterapkan teknik augmentasi data yang meliputi *random horizontal flip*, *random rotation*, dan *color jitter* untuk memberikan variasi pada *brightness*, *contrast*, *saturation*, dan *hue*. Proses ini bertujuan untuk meningkatkan *robustness* model terhadap berbagai kondisi pencahayaan dan

orientasi gestur. Kemudian seluruh citra dikonversi ke format RGB dan ditransformasi menjadi tensor PyTorch, sebelum dinormalisasi menggunakan parameter standar ImageNet dengan nilai *mean* [0.485, 0.456, 0.406] dan *standard deviation* [0.229, 0.224, 0.225]. Normalisasi ini memungkinkan pemanfaatan *transfer learning* dari model yang telah dilatih pada dataset ImageNet. Dataset kemudian dibagi menjadi tiga bagian yaitu data latih, data validasi, dan data uji.

2.3 Perancangan dan Pengembangan Model

Model yang digunakan dalam penelitian ini adalah varian arsitektur VGG seperti VGG-11, VGG-13, VGG-16, dan VGG-19. Arsitektur ini dipilih untuk mengetahui keandalan serta beban komputasi yang dibutuhkan dengan membandingkan keempat varian tersebut khususnya pada klasifikasi bahasa isyarat SIBI statis. Setiap varian akan dievaluasi dari segi efisiensi waktu pemrosesan dan metrik evaluasi untuk menentukan arsitektur yang paling optimal. Perbandingan spesifikasi keempat varian arsitektur VGG dapat dilihat pada Tabel 1. Perbedaan utama terletak pada jumlah lapisan konvolusi pada setiap blok, dimana VGG-11 memiliki struktur paling sederhana, sedangkan VGG-19 memiliki struktur paling kompleks dengan lapisan konvolusi terbanyak.

Tabel 1. Varian Arsitektur VGG [18]

VGG-11	VGG-13	VGG-16	VGG-19
Conv3-64	Conv3-64	Conv3-64	Conv3-64
	Conv3-64	Conv3-64	Conv3-64
Maxpool	Maxpool	Maxpool	Maxpool
Conv3-128	Conv3-128	Conv3-128	Conv3-128
	Conv3-128	Conv3-128	Conv3-128
Maxpool	Maxpool	Maxpool	Maxpool
Conv3-256	Conv3-256	Conv3-256	Conv3-256
Conv3-256	Conv3-256	Conv3-256	Conv3-256
		Conv3-256	Conv3-256
			Conv3-256
Maxpool	Maxpool	Maxpool	Maxpool
Conv3-512	Conv3-512	Conv3-512	Conv3-512
Conv3-512	Conv3-512	Conv3-512	Conv3-512
		Conv3-512	Conv3-512
			Conv3-512
Maxpool	Maxpool	Maxpool	Maxpool
Conv3-512	Conv3-512	Conv3-512	Conv3-512
Conv3-512	Conv3-512	Conv3-512	Conv3-512
		Conv3-512	Conv3-512
			Conv3-512
Maxpool	Maxpool	Maxpool	Maxpool
FC-4096	FC-4096	FC-4096	FC-4096
FC-4096	FC-4096	FC-4096	FC-4096
FC-1000	FC-1000	FC-1000	FC-1000
Soft-max	Soft-max	Soft-max	Soft-max

2.4 Pelatihan Model

Proses pelatihan model dilakukan selama 25 *epoch* menggunakan algoritma *Stochastic Gradient Descent* (SGD) dengan momentum 0.9 dan fungsi kehilangan *CrossEntropyLoss*. Penyesuaian *learning rate* diatur menggunakan *scheduler StepLR* untuk mengoptimalkan proses konvergensi. Setiap *epoch*, model dievaluasi terhadap data validasi untuk memantau performa, dan model terbaik disimpan berdasarkan akurasi validasi tertinggi. Seluruh metrik seperti akurasi, *loss*, dan perubahan *learning rate* dicatat untuk analisis performa selama proses pelatihan berlangsung.

2.5 Evaluasi

Evaluasi kinerja model dalam penelitian ini dilakukan secara komprehensif melalui implementasi beragam metrik evaluasi yang saling melengkapi untuk memberikan gambaran menyeluruh tentang kemampuan klasifikasi sistem. Metrik akurasi digunakan sebagai indikator global untuk mengukur proporsi prediksi yang benar terhadap total sampel, sementara *precision* dan *recall* dianalisis untuk memahami *trade-off* antara ketepatan prediksi positif dan kemampuan model dalam mendeteksi semua *instance* positif dari setiap kelas. *F1-score* diimplementasikan sebagai *harmonic mean* dari *precision* dan *recall* untuk memberikan ukuran seimbang yang mempertimbangkan kedua aspek tersebut, terutama penting dalam menangani dataset yang memiliki distribusi kelas tidak seimbang. *Mean class accuracy* dihitung untuk memastikan bahwa performa model tidak bias terhadap kelas mayoritas dan memberikan evaluasi yang adil untuk semua kategori. Selain metrik kuantitatif tersebut, *confusion matrix* dianalisis secara detail untuk mengidentifikasi pola spesifik kesalahan klasifikasi antarkelas, memungkinkan identifikasi kelas-kelas yang sering mengalami *misclassification* dan memberikan *insight* tentang kesamaan fitur visual yang menyebabkan kebingungan model. Protokol evaluasi ini diterapkan secara konsisten pada keempat varian model yang dikembangkan, menghasilkan empat set evaluasi komprehensif yang memungkinkan perbandingan sistematis kinerja relatif dan identifikasi varian optimal untuk implementasi praktis.

2.6 *Fine Tuning* dan Optimisasi

Tahap *fine tuning* dilakukan dengan membekukan sebagian lapisan awal pada arsitektur VGG untuk mempertahankan fitur dasar, sementara lapisan atas dilatih ulang agar lebih adaptif terhadap karakteristik data SIBI. Proses ini menggunakan algoritma optimisasi Adam dengan *learning rate* yang disesuaikan secara bertahap menggunakan *Cosine Annealing Scheduler* selama 15 *epoch*. Selama pelatihan, model dipantau berdasarkan akurasi validasi, dan parameter terbaik disimpan untuk evaluasi akhir. Strategi ini memungkinkan model untuk mengenali variasi bentuk tangan dan ekspresi wajah secara lebih akurat sambil mempertahankan kemampuan generalisasi yang baik.

3. Hasil dan Diskusi

Bagian ini menyajikan hasil implementasi dan evaluasi model penerjemah bahasa isyarat SIBI statis menggunakan empat varian arsitektur VGG. Pembahasan dimulai dari implementasi sistem, analisis proses pelatihan, hingga perbandingan performa setiap arsitektur berdasarkan berbagai metrik evaluasi dan beban komputasi model.

3.1. Implementasi Program

Implementasi sistem dilakukan menggunakan *framework PyTorch* dengan *Python3.8* melalui *Google Colab* dengan memanfaatkan *GPU Tesla T4*. Keempat varian arsitektur VGG diimplementasikan dengan perbedaan utama pada kedalaman *convolutional layers*, sementara struktur *classifier* tetap konsisten untuk memastikan *fair comparison*. Setiap model VGG terdiri dari dua komponen utama yaitu *feature extraction layers* (*convolutional*) dan *classifier layers* (*fully connected*). Perbedaan antar varian terletak pada jumlah *convolutional layers* dalam *feature extraction*, yang secara progresif meningkat dari VGG-11 hingga VGG-19.

Tabel 2 sampai Tabel 5 menunjukani implementasi *progression* yang sistematis dalam kedalaman arsitektur yaitu, VGG-11 (8 *conv layers*), VGG-13 (10 *conv layers*), VGG-16 (13 *conv layers*), dan VGG-19 (16 *conv layers*), dimana setiap peningkatan kedalaman dirancang untuk meningkatkan *representational capacity* model. Untuk memastikan *fair comparison*, semua varian menggunakan *classifier architecture* yang identik dengan dua *hidden layers* (4096 *neurons each*) dan *dropout rate* 0.5, sehingga perbedaan performa yang dihasilkan murni berasal dari *feature extraction capability* yang berbeda. Final layer pada semua model disesuaikan untuk mengakomodasi 10 kelas SIBI (num_classes=10), menggantikan konfigurasi original ImageNet yang memiliki 1000 kelas.

p-ISSN: 2986-3929
e-ISSN: 3032-1948

Tabel 2. Implementasi Kode Program Model VGG-11

```
class VGG11(nn.Module):
    def __init__(self, num_classes=1000):
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(64, 128, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(128, 256, 3, padding=1), nn.ReLU(),
            nn.Conv2d(256, 256, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(256, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.classifier = nn.Sequential(
            nn.AdaptiveAvgPool2d((7, 7)), nn.Flatten(),
            nn.Linear(25088, 4096), nn.ReLU(), nn.Dropout(0.5),
            nn.Linear(4096, 4096), nn.ReLU(), nn.Dropout(0.5),
            nn.Linear(4096, num_classes)
        )

    def forward(self, x):
        return self.classifier(self.features(x))
```

Tabel 3. Implementasi Kode Program Model VGG-13

```
class VGG13(nn.Module):
    def __init__(self, num_classes=1000):
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, 3, padding=1), nn.ReLU(),
            nn.Conv2d(64, 64, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(64, 128, 3, padding=1), nn.ReLU(),
            nn.Conv2d(128, 128, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(128, 256, 3, padding=1), nn.ReLU(),
            nn.Conv2d(256, 256, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(256, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.classifier = nn.Sequential(
            nn.AdaptiveAvgPool2d((7, 7)), nn.Flatten(),
            nn.Linear(25088, 4096), nn.ReLU(), nn.Dropout(0.5),
```

p-ISSN: 2986-3929
e-ISSN: 3032-1948

```
            nn.Linear(4096, 4096), nn.ReLU(), nn.Dropout(0.5),
            nn.Linear(4096, num_classes)
        )

    def forward(self, x):
        return self.classifier(self.features(x))
```

Tabel 4. Model VGG-16

```
class VGG16(nn.Module):
    def __init__(self, num_classes=1000):
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, 3, padding=1), nn.ReLU(),
            nn.Conv2d(64, 64, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(64, 128, 3, padding=1), nn.ReLU(),
            nn.Conv2d(128, 128, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(128, 256, 3, padding=1), nn.ReLU(),
            nn.Conv2d(256, 256, 3, padding=1), nn.ReLU(),
            nn.Conv2d(256, 256, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(256, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.classifier = nn.Sequential(
            nn.AdaptiveAvgPool2d((7, 7)), nn.Flatten(),
            nn.Linear(25088, 4096), nn.ReLU(), nn.Dropout(0.5),
            nn.Linear(4096, 4096), nn.ReLU(), nn.Dropout(0.5),
            nn.Linear(4096, num_classes)
        )

    def forward(self, x):
        return self.classifier(self.features(x))
```

Tabel 5. Model VGG-19

```
class VGG19(nn.Module):
    def __init__(self, num_classes=1000):
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, 3, padding=1), nn.ReLU(),
            nn.Conv2d(64, 64, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(64, 128, 3, padding=1), nn.ReLU(),
            nn.Conv2d(128, 128, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
```

p-ISSN: 2986-3929
e-ISSN: 3032-1948

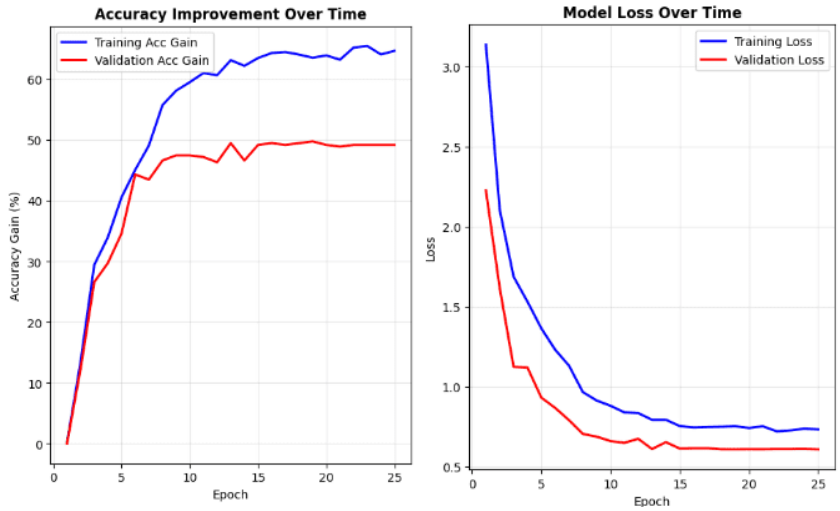
```
            nn.Conv2d(128, 256, 3, padding=1), nn.ReLU(),
            nn.Conv2d(256, 256, 3, padding=1), nn.ReLU(),
            nn.Conv2d(256, 256, 3, padding=1), nn.ReLU(),
            nn.Conv2d(256, 256, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(256, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.classifier = nn.Sequential(
            nn.AdaptiveAvgPool2d((7, 7)), nn.Flatten(),
            nn.Linear(25088, 4096), nn.ReLU(), nn.Dropout(0.5),
            nn.Linear(4096, 4096), nn.ReLU(), nn.Dropout(0.5),
            nn.Linear(4096, num_classes)
        )

    def forward(self, x):
        return self.classifier(self.features(x))
```

3.2. Hasil Pelatihan Model

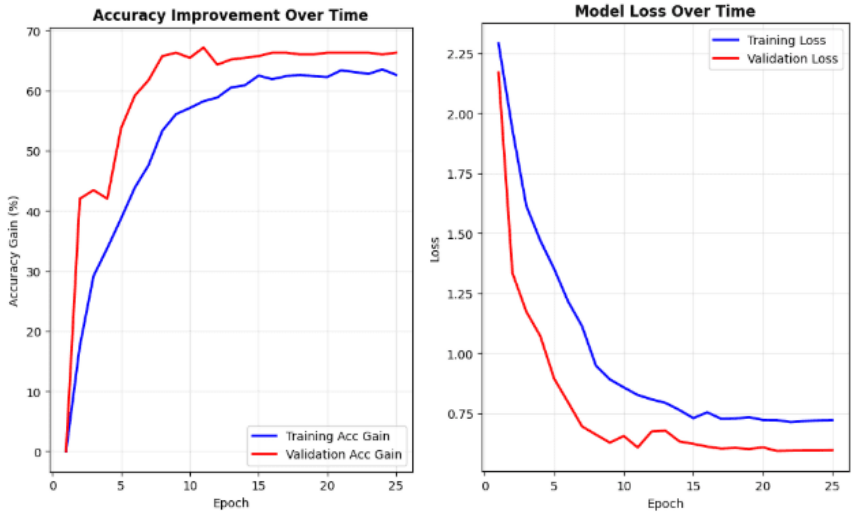
Eksperimen pelatihan dilakukan dengan protokol standar 25 *epoch* untuk seluruh varian arsitektur VGG, dengan pemantauan berkelanjutan terhadap perkembangan akurasi dan fungsi *loss* sebagaimana ditampilkan pada Gambar 3. Analisis kurva pembelajaran menunjukkan pola konvergensi yang berbeda-beda antar varian, mengindikasikan bahwa kedalaman arsitektur memiliki pengaruh signifikan terhadap dinamika pembelajaran dan kemampuan generalisasi model. Hasil observasi empiris menunjukkan bahwa peningkatan kompleksitas arsitektur dari VGG-11 hingga VGG-16 menghasilkan perbaikan progresif dalam performa validasi, dengan akurasi pelatihan yang meningkat secara konsisten dari 74.33% (VGG-11) menjadi 78.07% (VGG-16), disertai penurunan substansial dalam *validation loss* dari 0.6079 menjadi 0.5041. Tren ini mengkonfirmasi ekspektasi teoritis bahwa peningkatan kedalaman jaringan dapat meningkatkan kapasitas representasi fitur hingga mencapai titik optimal tertentu.

VGG-16 mendemonstrasikan performa paling seimbang dengan mencapai akurasi pelatihan 78.07% dan akurasi validasi 81.14%, mengindikasikan keseimbangan optimal antara kompleksitas model dan kemampuan generalisasi untuk dataset SIBI statis yang digunakan. Sebaliknya, VGG-19 menunjukkan fenomena menarik dimana akurasi pelatihan menurun menjadi 76.61% namun akurasi validasi mencapai nilai tertinggi 82.29%, menunjukkan tanda-tanda awal perilaku *overfitting* dimana model yang terlalu kompleks mulai menghafal pola pelatihan daripada mempelajari fitur yang dapat digeneralisasi. Perbedaan antara metrik pelatihan dan validasi pada VGG-19 mengkonfirmasi bahwa kompleksitas arsitektur yang berlebihan dapat menjadi kontraproduktif untuk dataset dengan keragaman terbatas, dimana jaringan yang lebih sederhana dengan regularisasi yang tepat dapat mencapai performa generalisasi yang superior. Temuan ini memberikan wawasan penting tentang kriteria pemilihan arsitektur untuk aplikasi domain spesifik dalam tugas *computer vision*.

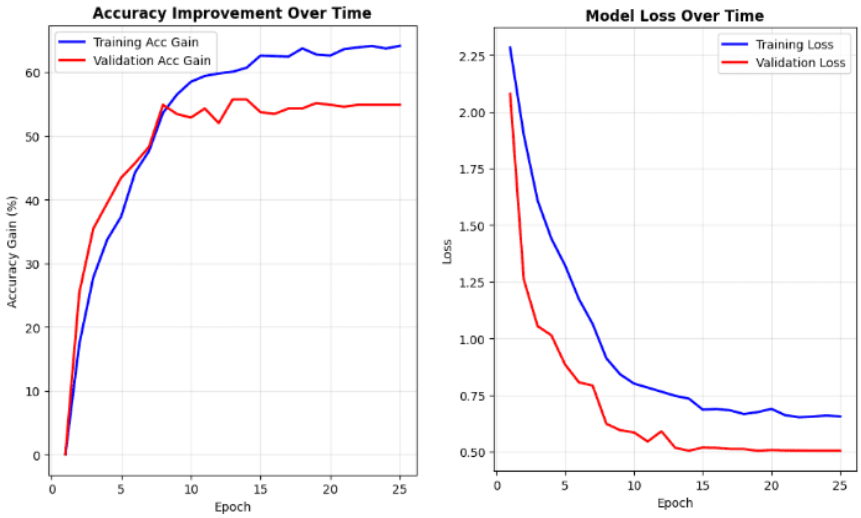
p-ISSN: 2986-3929
e-ISSN: 3032-1948



(a)

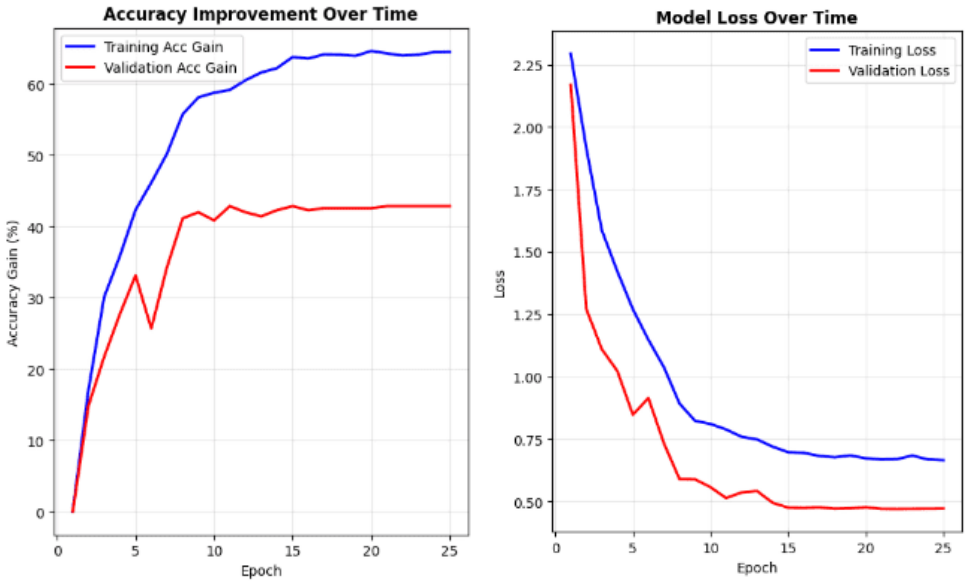


(b)



(c)

p-ISSN: 2986-3929
e-ISSN: 3032-1948



(d)

Gambar 3. Hasil Perbandingan Akurasi dan *Loss* Setiap Model (a) VGG-11, (b) VGG-13, (c) VGG-16, (d) VGG-19

3.3. Evaluasi Performa Arsitektur VGG

a. Metrik Evaluasi Komprehensif

Tabel 6 menampilkan perbandingan metrik evaluasi untuk keempat varian arsitektur VGG pada dataset *testing*:

Tabel 6. Perbandingan Metrik Evaluasi Arsitektur VGG

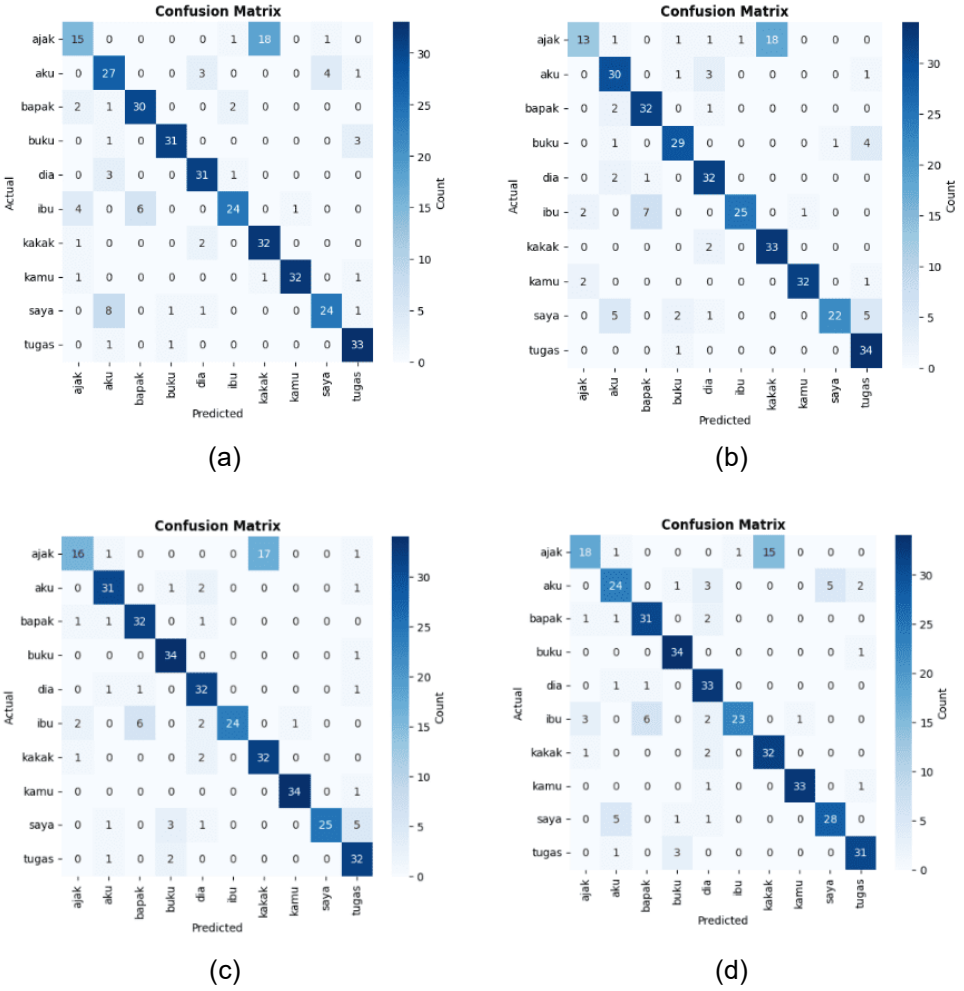
Arsitektur	Akurasi (%)	Precision (%)	Recall (%)	F1-Score (%)
VGG-11	79.7	80.5	79.7	79.3
VGG-13	80.6	82.4	80.6	79.7
VGG-16	83.4	85.2	83.4	82.9
VGG-19	82.0	82.9	82.0	81.6

VGG-16 menunjukkan performa terbaik di semua metrik evaluasi, dengan akurasi mencapai 92.86%. Peningkatan performa yang signifikan terlihat dari VGG-11 ke VGG-16, namun VGG-19 mengalami sedikit penurunan dibandingkan VGG-16, yang mengindikasikan adanya *diminishing returns* pada arsitektur yang terlalu dalam untuk dataset ini.

b. Analisis *Confusion Matrix*

Confusion matrix memberikan *insight* mendalam tentang performa klasifikasi setiap model untuk 10 kelas kata SIBI yang dapat dilihat pada Gambar 4-7.

p-ISSN: 2986-3929
e-ISSN: 3032-1948



Gambar 4. *Confusion Matrix* (a) VGG-11, (b) VGG-13, (c) VGG-16, (d) VGG-19

Confusion matrix memberikan gambaran yang jelas tentang bagaimana setiap model mengenali 10 kelas kata SIBI, sebagaimana terlihat pada Gambar 4 hingga Gambar 7. Hal menarik adalah adanya pola kesalahan yang konsisten di semua arsitektur. Kata "ajak" ternyata menjadi tantangan terbesar bagi semua model, dengan tingkat keberhasilan hanya berkisar 37-48% karena sering keliru dikenali sebagai "kakak" (15-18 kasus kesalahan di setiap model). Di sisi lain, kata-kata seperti "kakak", "kamu", dan "tugas" cenderung mudah dikenali dengan akurasi di atas 90%. Hal yang juga menarik perhatian adalah kesalahan konsisten dalam mengenali "ibu" yang sering dikira "bapak" (6-7 kasus di semua model), menunjukkan bahwa gerakan isyarat untuk anggota keluarga memang memiliki kemiripan yang membingungkan.

Ketika melihat perkembangan dari model sederhana ke kompleks, terlihat cerita yang menarik. VGG-11 memberikan fondasi dasar dengan rentang akurasi 43-94%, lalu VGG-13 menunjukkan peningkatan pada beberapa kata seperti "tugas" yang mencapai 97%, meski "ajak" justru makin sulit dikenali (turun ke 37%). VGG-16 berhasil menemukan keseimbangan terbaik dengan pemulihan signifikan pada "buku" dan "kamu" yang mencapai 97%, plus sedikit perbaikan pada "ajak" menjadi 46%. Sementara VGG-19, meski lebih kompleks, memberikan hasil yang campur aduk, ada perbaikan pada "ajak" (49%) dan "saya" (80%), namun tidak konsisten di setiap kelas yang ada. Dari sini terlihat bahwa VGG-16 memberikan performa paling seimbang untuk dataset ini. Fenomena ini mengkonfirmasi prinsip *bias-variance trade-off* dimana model dengan kompleksitas optimal dapat mencapai generalisasi terbaik tanpa mengalami *underfitting* maupun

overfitting yang ekstrem.

3.4. Analisis Kompleksitas Komputasi

Selain mengevaluasi akurasi model, penting untuk menganalisis kompleksitas komputasi yang diperlukan oleh setiap arsitektur. Analisis ini memberikan perspektif tentang *trade-off* antara performa akurasi dan *efisiensi resource*, yang menjadi pertimbangan krusial dalam implementasi praktis sistem pengenalan bahasa isyarat SIBI.

a. Perbandingan GFLOPs dan Parameter Model

Analisis kompleksitas komputasi menggunakan metrik GFLOPs memberikan gambaran tentang beban komputasi yang diperlukan setiap arsitektur untuk melakukan inferensi pada input image 224×224 piksel. Perhitungan *Efficiency Score* diperoleh dari akurasi dibagi GFLOPs, dimana nilai yang lebih tinggi mengindikasikan efisiensi komputasional yang lebih baik. Hasil perbandingan dapat dilihat pada Tabel 7.

Tabel 7. GPLOPs setiap Model Serta Tingat Efisensi Arsitektur VGG

Arsitektur	GFLOPs	Parameters(M)	Model Size	Akurasi Test (%)	Efisiensi Score
VGG-11	7.62	132.86	506.8	79.7	10.46
VGG-13	11.32	128.99	492.1	80.6	7.12
VGG-16	15.48	134.30	512.3	83.4	5.39
VGG-19	19.64	139.61	532.6	82.0	4.18

b. *Trade-off* Antara Kompleksitas dan Performa

Hasil analisis menunjukkan adanya pola yang jelas dalam hubungan antara kompleksitas komputasi dan akurasi model. VGG-11 memberikan efisiensi tertinggi dengan skor 10.46, mencapai akurasi 79.7% dengan beban komputasi terendah (7.62 GFLOPs). Hal ini menunjukkan bahwa arsitektur sederhana dapat memberikan nilai efisiensi yang optimal untuk aplikasi dengan keterbatasan resource.

Seiring peningkatan kedalaman arsitektur, terjadi kenaikan GFLOPs yang signifikan: VGG-13 memerlukan 11.32 GFLOPs (peningkatan 48.6%), VGG-16 membutuhkan 15.48 GFLOPs (peningkatan 36.7% dari VGG-13), hingga VGG-19 yang mencapai 19.64 GFLOPs (peningkatan 26.9% dari VGG-16). Namun, peningkatan akurasi tidak sebanding dengan peningkatan kompleksitas komputasi yang terjadi.

VGG-16 mencapai performa akurasi terbaik (83.4%) dengan efisiensi score 5.39, menunjukkan keseimbangan yang masih dapat diterima antara akurasi dan kompleksitas. Sebaliknya, VGG-19 mengalami fenomena *diminishing returns*, dimana peningkatan kompleksitas 26.9% justru menghasilkan penurunan akurasi menjadi 82.0%, sehingga efisiensi score turun menjadi 4.18. Fenomena ini mengkonfirmasi bahwa arsitektur yang terlalu kompleks tidak selalu memberikan hasil yang lebih baik.

c. Distribusi Beban Komputasi dalam Arsitektur

Analisis *breakdown* GFLOPs mengungkap pola yang konsisten di semua arsitektur, dimana 98-99% komputasi terjadi pada *feature extraction layers* (*convolutional layers*), sementara *classifier layers* (*fully connected*) hanya berkontribusi 0.6-1.6% dari total beban komputasi. VGG-11 menunjukkan distribusi 7.49 GFLOPs (98.4%) untuk convolution dan 0.12 GFLOPs (1.6%) untuk *fully connected layers*. Pattern serupa berlanjut hingga VGG-19 dengan 19.52 GFLOPs (99.4%) dan 0.12 GFLOPs (0.6%) masing-masing.

Penelitian ini memiliki implikasi penting untuk optimasi performa, dimana fokus pengembangan dan perbaikan efisiensi harus diarahkan pada *convolutional layers rather than classifier layers*. Konsistensi *pattern* ini *across all architectures* menunjukkan bahwa karakteristik ini merupakan sifat intrinsik dari arsitektur VGG.

4. Kesimpulan

Penelitian ini berhasil mengidentifikasi VGG-16 sebagai arsitektur optimal untuk klasifikasi bahasa isyarat SIBI statis melalui evaluasi komprehensif terhadap empat varian arsitektur VGG. VGG-16 menunjukkan performa superior dengan akurasi 83.4%, *precision* 85.2%, *recall* 83.4%, dan *F1-score* 82.9%, mencapai keseimbangan ideal antara kompleksitas model dan kemampuan generalisasi. Fenomena *diminishing returns* terkonfirmasi pada VGG-19 yang justru mengalami penurunan performa menjadi 82.0% meskipun memiliki arsitektur lebih kompleks, membuktikan bahwa peningkatan kedalaman layer tidak selalu berkorelasi positif dengan peningkatan akurasi. Analisis *trade-off* kompleksitas komputasi menunjukkan bahwa meskipun VGG-11 memiliki efisiensi tertinggi (10.46 GFLOPs), VGG-16 tetap memberikan nilai optimal dengan mempertimbangkan kebutuhan akurasi tinggi untuk aplikasi praktis.

Hasil di dapat memberikan kontribusi signifikan dalam mengatasi gap penelitian sebelumnya dan menyediakan *foundation solid* untuk pengembangan teknologi *assistive* bagi komunitas tunarungu di Indonesia. Identifikasi tantangan spesifik seperti kesulitan membedakan gestur serupa ("ajak" vs "kakak", "ibu" vs "bapak") memberikan *valuable insight* untuk perbaikan dataset dan metodologi di masa depan. Hasil penelitian ini secara langsung mendukung pengembangan sistem penerjemah bahasa isyarat SIBI yang lebih efektif dan efisien, berkontribusi dalam menjembatani kesenjangan komunikasi antara penyandang tunarungu dengan masyarakat umum, sekaligus memajukan implementasi teknologi *computer vision* untuk aplikasi sosial yang bermakna.

Daftar Pustaka

[1] A. Alamsyah and D. A. Anggraeni, “Detection of Indonesian Sign Language System using Convolutional Neural Network (CNN) with Nadam Optimizer,” 2024, pp. 352–359. doi: 10.2991/978-94-6463-589-8_32.

[2] I. A. Adeyanju, O. O. Bello, and M. A. Adegboye, “Machine learning methods for sign language recognition: A critical review and analysis,” Nov. 01, 2021, *Elsevier B.V.* doi: 10.1016/j.iswa.2021.200056.

[3] E. Libra Kelana, M. Riko, A. Prasetya, and M. Zulfadhilah, “Integrating the CNN Model with the Web for Indonesian Sign Language (BISINDO) Recognition,” 2025. [Online]. Available: http://jurnal.polibatam.ac.id/index.php/JAIC

[4] A. C. R. Lorentzen, “Digital transformation as distributed leadership: Firing the change agent,” in *Procedia Computer Science*, Elsevier B.V., 2021, pp. 245–254. doi: 10.1016/j.procs.2021.12.011.

[5] P. W. Aditama, P. S. U. Putra, I. M. M. Yusa, and I. N. T. A. Putra, “Designing augmented reality sibi sign language as a learning media,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Mar. 2021. doi: 10.1088/1742-6596/1810/1/012038.

[6] M. Sholawati, K. Auliasari, and F. X. Ariwibisono, “Pengembangan Aplikasi Pengenalan Bahasa Isyarat Abjad Sibi Menggunakan Metode Convolutional Neural Network (CNN),” 2022.

[7] S. Syamsuddin, T. Pristiwaluyo, W. A Saleh, and Z. Zulfitrah, “Development of Indonesian Sign Language System (SIBI) Dictionary Application for Students with Special Needs,” *AL-ISHLAH: Jurnal Pendidikan*, vol. 15, no. 3, pp. 3132–3143, Sep. 2023, doi: 10.35445/alishlah.v15i3.3529.

[8] R. Rastgoo, K. Kiani, and S. Escalera, “Sign Language Recognition: A Deep Survey,” Feb. 01, 2021, *Elsevier Ltd*. doi: 10.1016/j.eswa.2020.113794.

[9] I. T. Ahmed, W. H. Gwad, B. T. Hammad, and E. Alkayal, “Enhancing Hand Gesture Image Recognition by Integrating Various Feature Groups,” *Technologies (Basel)*, vol. 13, no. 4, Apr. 2025, doi: 10.3390/technologies13040164.

[10] M. E. Al Rivan and S. Hartoyo, “Klasifikasi Isyarat Bahasa Indonesia Menggunakan Metode Convolutional Neural Network,” *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 8, no. 2, Aug. 2022, doi: 10.28932/jutisi.v8i2.4863.
[11] A. Muh. A. Siddik, “Comparison of Transfer Learning Algorithm Performance in Hand Sign Language Digits Image Classification,” *Jurnal Matematika, Statistika dan Komputasi*, vol. 20, no. 1, pp. 75–89, Sep. 2023, doi: 10.20956/j.v20i1.26503.
[12] E. Strago Giamiko, E. Lesmana Tjiong, and J. Pulomas Selatan Kav, “Pengembangan Aplikasi Pengenalan Tulisan Tangan Abjad dan Angka Berbasis Convolutional Neural Network,” 2024.
[13] S. Nurdiati *et al.*, “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.”
[14] R. A. Tilasefana and R. E. Putra, “Penerapan Metode Deep Learning Menggunakan Algoritma CNN Dengan Arsitektur VGG NET Untuk Pengenalan Cuaca,” *Journal of Informatics and Computer Science*, vol. 05, 2023.
[15] Weny Indah Kusumawati and Adisaputra Zidha Noorizki, “Perbandingan Performa Algoritma VGG16 Dan VGG19 Melalui Metode CNN Untuk Klasifikasi Varietas Beras,” *Journal of Computer, Electronic, and Telecommunication*, vol. 4, no. 2, Dec. 2023, doi: 10.52435/complete.v4i2.387.
[16] M. D. Meitantya, C. A. Sari, E. H. Rachmawanto, and R. R. Ali, “VGG-16 ARCHITECTURE ON CNN FOR AMERICAN SIGN LANGUAGE CLASSIFICATION,” *Jurnal Teknik Informatika (Jutif)*, vol. 5, no. 4, pp. 1165–1171, Jul. 2024, doi: 10.52436/1.jutif.2024.5.4.2160.
[17] I. D. M. B. A. Darmawan, Linawati, G. Sukadarmika, N. M. A. E. D. Wirastuti, and R. Pulungan, “Indonesian sign language system (SIBI) dataset: Sentences enhanced by diverse facial expressions for total communication,” *Data Brief*, vol. 60, Jun. 2025, doi: 10.1016/j.dib.2025.111642.
[18] D. Sumarlie, C. Lubis, and T. Handhayani, “Pengenalan Kue Tradisional Indonesia Menggunakan Algoritma Convolutional Neural Network,” 2022. [Online]. Available: https://arxiv.org/abs/1409.1556,