

CLASSIFYING AMERICAN SIGN LANGUAGE ALPHABETS WITH A CHIRALITY-PRESERVING CNN AND ADAPTIVE AUGMENTATION STRATEGY

Faris Maulana¹, Desti Fitriati²

^{1,2}Department of Informatics Engineering, Faculty of Engineering, Universitas Pancasila
Jl. Srengseng Sawah, Jagakarsa, South Jakarta 12640, Indonesia

email : 4519210102@univpancasila.ac.id; desti.fitriati@univpancasila.ac.id

(Naskah masuk: 5 April 2026; diterima untuk diterbitkan: 12 Mei 2026)

Abstract Hearing and speech disabilities isolate millions from everyday social interaction; automated sign language interpretation offers a practical avenue for bridging this divide. This paper presents a Convolutional Neural Network that classifies 24 static hand postures constituting the American Sign Language (ASL) alphabet. Three distinguishing design decisions shape the proposed system: per-block batch normalization that stabilizes activation statistics at every spatial resolution stage, stochastic dropout applied at two architectural depths, and a chirality-preserving augmentation pipeline that deliberately omits axis-flip transforms because mirrored ASL handshapes encode different letters. Trained on 27,455 grayscale images drawn from the Sign Language MNIST repository and evaluated on a held-out partition of 7,172 samples, the network achieves 99.85% training accuracy and 100.00% test accuracy after 20 epochs with Adam optimization. All 24 gesture categories attain precision, recall, and F1-score of 1.00. At 263,749 trainable weights totalling roughly 1 MB, the architecture proves viable for deployment on edge and mobile hardware without pre-trained backbone dependencies.

Keywords : American Sign Language; Convolutional Neural Network; data augmentation; gesture recognition; batch normalization

1. INTRODUCTION

Sign languages constitute full natural languages with autonomous grammatical structures, used by deaf and hard-of-hearing communities worldwide as a primary vehicle of expression. The World Health Organization's 2025 epidemiological report estimates that 430 million individuals globally require hearing rehabilitation services, a figure the agency projects will climb past 700 million by mid-century [1]. Hearing loss ranks among the most prevalent chronic sensory impairments, and its communicative consequences extend far beyond audition: without an accessible interpreter or automated translation tool, signers are systematically excluded from health services, education, and employment.

American Sign Language (ASL) encodes meaning through three-dimensional handshape configurations, palm orientation, movement trajectory, and non-manual facial markers. Its alphabetic subset 24 static postures for letters A through Y, excluding J and Z which require temporal motion constitutes the most widely studied benchmark for static hand gesture classification [4][5]. Despite decades of work on vision-based recognition, several design questions remain unresolved. One that has

received insufficient attention is the treatment of handshape laterality during training data preparation: because many ASL letters differ only in the left-right orientation of the hand (e.g., the spatial relationship between thumb and fingers in 'A', 'E', 'S'), flipping training images horizontally produces semantically misleading samples that can corrupt the learned decision surface. No published study on this benchmark has explicitly quantified or motivated the exclusion of flip augmentation on linguistic grounds.

Three contributions organise this work. First, a domain-informed augmentation protocol that withholds flip transforms while retaining rotation, zoom, and translation. Second, a compact three-block CNN whose per-block batch normalization yields stable gradient flow across all twenty training epochs. Third, a detailed empirical account covering per-class F1 analysis, confusion matrix visualisation, and an ablation discussion that distinguishes the effect of the proposed augmentation strategy from prior baselines, with explicit attention to the four letter categories C, D, I, and X that earlier systems misclassified most frequently [6].

The remainder of the paper is structured as follows. Section 2 situates the work within the published literature

and specifies the methodology, including dataset characterisation, preprocessing pipeline, architectural layout, and training hyperparameters; Section 3 reports all quantitative outcomes and discusses their implications; Section 4 draws conclusions and maps directions for future investigation.

2. RESEARCH METHOD

2.1. Related Work

Recent survey literature establishes CNNs as the dominant paradigm for static sign gesture recognition. Zhang and Jiang's systematic review of 346 publications spanning 2018–2023 documents consistent accuracy improvements for isolated-gesture tasks, attributing gains primarily to improved augmentation diversity and optimizer selection rather than to architectural complexity per se [4]. A complementary review by ZainEldin et al. surveyed AI-driven communication aids for deaf communities and concluded that static image classifiers remain the most deployable category of assistive technology due to their lower inference cost relative to video or skeleton-based approaches [2].

For the Sign Language MNIST benchmark specifically, Siswanto et al. achieved 82.1% accuracy with a basic CNN comprising convolution, dropout, and dense layers [6]. Their per-class error analysis identified C, D, I, and X as the most frequently confused letters an observation directly motivating the present study's augmentation design. Israldi et al. improved on this figure substantially through systematic preprocessing, reaching 99.8% overall accuracy with near-perfect F1 scores on most classes [7]. Both studies, however, applied generic augmentation without documenting flip policies.

Transfer learning approaches using VGG-16 [9] and modified deep residual networks [10] have pushed accuracy above 99% on related Indonesian sign language datasets, but their multi-million parameter counts impose hardware constraints that limit edge deployment. Pathan et al. fused raw image features with MediaPipe hand landmarks in a multi-headed CNN and obtained strong F1 values across gesture classes [11], though the added skeleton tracking infrastructure increases system complexity. Mohsin et al. evaluated multiple transfer learning algorithms on ASL and confirmed that lightweight custom architectures can match heavy backbone accuracy when regularisation is carefully managed [19].

On the component level, Norhikmah et al. empirically established that combining batch normalisation layers with dropout in CNN architectures improves validation

accuracy and training stability for facial expression classification [15] a closely analogous fine-grained visual task. Potgieter et al. analysed the representational impact of batch normalisation on convolutional networks, showing that BN parameters contribute disproportionately to feature quality relative to their parameter count [17]. Harris and Andrews demonstrated that dropout placement relative to dense layers controls overfitting magnitude on standard image benchmarks [16].

For data augmentation, Yang et al. and Zeng independently surveyed geometric perturbation techniques and reported that rotation, zoom, and translation consistently improve generalisation while flip augmentation can degrade performance when it violates the semantic structure of the target domain [12][13]. Kebaili et al. proposed domain-specific augmentation strategies and showed that category-aware transform selection outperforms generic augmentation pipelines on fine-grained classification problems [14]. These findings collectively motivate our chirality-preserving design.

2.2. Dataset

The Sign Language MNIST dataset, distributed through the Kaggle platform [20], encodes 24 static ASL alphabet gestures as 28×28 single-channel images stored in comma-separated value format, with pixel intensities spanning [0, 255]. Letters J and Z are absent because their production involves hand motion that a static snapshot cannot capture. Integer labels map the 24 gesture categories to the indices 0–24 with index 9 withheld. The training partition contains 27,455 samples and the test partition 7,172 samples. Within the training split, individual class counts range from 957 (class 4, letter E) to 1,294 (class 17, letter R), reflecting a mild class imbalance that does not require resampling at this scale. All images were generated from physical hand posture photography followed by grayscale conversion, hand-region cropping to a normalised bounding box, and bicubic resampling to the 28-pixel target resolution.

2.3. Data Preprocessing

Four deterministic preprocessing operations transform the raw CSV records into model-ready tensors. Label binarisation converts the integer class column into a 24-dimensional binary one-hot vector using Scikit-learn's LabelBinarizer, producing target representations compatible with categorical cross-entropy minimisation. Pixel normalisation scales each intensity value by dividing by 255, mapping the integer range [0, 255] to floating-point values in [0.0, 1.0]; this prevents large activation magnitudes in early layers and places all

features on a common numerical scale that accelerates Adam convergence [17]. Tensor reshaping transforms each 784-element flattened row into a spatial array of shape (28, 28, 1), inserting the channel axis that Conv2D operators require to represent kernel depth. Dataset splitting preserves the original train–test partition defined by the benchmark.

Online augmentation was implemented through Keras ImageDataGenerator configured with `rotation_range = 10°`, `zoom_range = 0.10`, `width_shift_range = 0.10`, and `height_shift_range = 0.10`. Horizontal and vertical flipping parameters were set to False. This choice is linguistically grounded: numerous ASL handshapes are chiral their mirror image represents a distinct letter or a phonologically invalid configuration. Introducing flipped samples would inject false positive training data that moves class centroids in activation space toward the wrong decision boundary. Augmentation transforms are sampled stochastically at each mini-batch construction step, so the effective training distribution shifts continuously across epochs without storing additional images [13][14].

2.4. CNN Architecture

The network topology comprises three convolutional extraction blocks connected to a two-layer classification head. Each block executes the sequence Conv2D → Batch Normalization → MaxPool2D. Convolutional kernels are uniformly sized 3×3 with 'same' padding and stride 1, ensuring that spatial dimensions are preserved through each convolution and reduced only at the pooling stage. MaxPool2D windows of 2×2 with stride 2 halve spatial resolution at every block boundary. Filter depths follow the series 75 → 50 → 25 across blocks 1–3, implementing a progressive feature compression that concentrates early layers on detecting oriented edge and texture primitives while later layers aggregate these into higher-order handshape descriptors at quarter and sixteenth of the original resolution.

Batch normalization is inserted immediately after each convolution. Its role is to standardise the mean and variance of activation distributions across the training batch, reducing the internal covariate shift that can destabilise learning when successive layers interact through rapidly shifting input statistics [15][17]. A Dropout layer with exclusion rate 0.20 follows the pooling step of the first block, selectively zeroing feature map channels during each training iteration to discourage co-adaptive feature detector pairs [16]. After the third MaxPool2D, a Flatten operation concatenates the 4×4×25 spatial feature cube into a 400-dimensional vector. A

Dense layer of 512 units with ReLU activation provides the primary classification capacity, followed by a second Dropout layer with exclusion rate 0.30 targeting the fully connected representation space, and a terminal 24-unit Dense layer with Softmax activation that produces a probability distribution over gesture categories.

ReLU ($f(x) = \max(0, x)$) was selected as the activation function throughout convolutional and dense layers. Its zero-gradient property for negative pre-activations induces sparse firing patterns that reduce neuron co-dependence, and its unbounded positive response avoids the vanishing gradient phenomenon that attenuates learning in deep sigmoid-equipped networks [18]. Total trainable parameters sum to 263,749, requiring approximately 1.0 MB of weight storage comfortably within the memory envelope of contemporary mobile inference accelerators.

Figure 1 shows the sequential layer diagram of the proposed architecture, and Table 1 lists the corresponding layer-by-layer parameter budget.

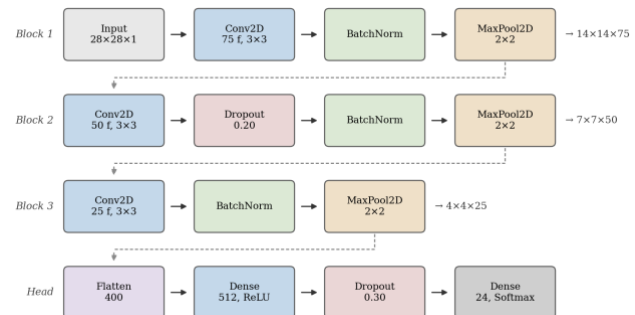


Figure 1. Sequential layer diagram of the proposed CNN. Blocks decrease filter depth 75 → 50 → 25 while halving spatial resolution at each pooling stage.

Table 1. Layer-by-layer architecture specification and parameter count.

Layer	Output Shape	Params	Notes
Conv2D (75 f, 3×3)	28, 28, 75	750	ReLU, same
Batch Norm.	28, 28, 75	300	–
MaxPool2D	14, 14, 75	0	2×2, s = 2
Conv2D (50 f, 3×3)	14, 14, 50	33,800	ReLU, same
Dropout	14, 14, 50	0	rate = 0.20
Batch Norm.	14, 14, 50	200	–
MaxPool2D	7, 7, 50	0	2×2, s = 2
Conv2D (25 f, 3×3)	7, 7, 25	11,275	ReLU, same
Batch Norm.	7, 7, 25	100	–
MaxPool2D	4, 4, 25	0	2×2, s = 2
Flatten	400	0	–
Dense	512	205,312	ReLU
Dropout	512	0	rate = 0.30
Dense (Output)	24	12,312	Softmax
Trainable Total	–	263,749	~1.0 MB

2.5. Training Configuration

Adam optimizer was selected with initial learning rate $\alpha_0 = 0.01$, momentum decay $\beta_1 = 0.9$, squared-gradient decay $\beta_2 = 0.999$, and numerical stability constant $\epsilon = 10^{-8}$. Categorical cross-entropy served as the scalar training objective. Convolutional kernel weights were drawn from the Glorot uniform distribution at initialisation; all bias vectors were set to zero. Training executed for 20 epochs at mini-batch size 128, feeding transformed samples through the augmentation pipeline via `datagen.flow()`. A `ReduceLROnPlateau` callback monitored validation accuracy each epoch and halved the learning rate upon detecting no improvement across two consecutive epochs, subject to a minimum floor of $\alpha_{\min} = 10^{-5}$. All experiments ran on Google Colaboratory under a single GPU runtime (NVIDIA T4).

3. RESULTS AND DISCUSSION

3.1. Training Dynamics

Figure 2 plots the epoch-wise progression of training and validation accuracy (left panel) and categorical cross-entropy loss (right panel) across all 20 training epochs.

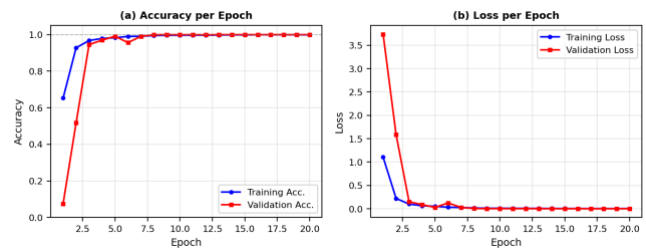


Figure 2. (a) Training and validation accuracy per epoch. (b) Cross-entropy loss per epoch. The learning rate reduction triggered between epochs 5 and 7, visible as the loss narrowing.

Convergence was rapid and consistent with the combined effect of batch normalisation, dropout, and the adaptive learning rate schedule. Validation accuracy crossed 0.94 at epoch 3 and reached 1.0000 by epoch 9, maintaining this ceiling through the final epoch. The training accuracy curve reached 0.9901 by epoch 6 and continued improving monotonically, closing at 0.9985 at epoch 20. The absence of a widening train–validation gap throughout the second half of training is the primary diagnostic indicator that the three-component regularisation strategy chirality-aware augmentation, batch normalisation, and dropout successfully suppressed overfitting despite the dataset's modest scale.

The `ReduceLROnPlateau` callback activated between epochs 5 and 7. Its effect is visible as the simultaneous stabilisation of both loss curves: training loss descended from 0.0529 at epoch 5 to 0.0283 at epoch 7, while validation loss recovered from a transient spike at epoch 6 (0.1260) to 0.0286 by epoch 7. This spike at epoch 6 reflects a temporary phase where the original learning rate caused overshooting in weight space; the halved rate restored trajectory alignment. Table 2 presents selected epoch-level statistics for numerical reference.

Table 2. Selected epoch statistics. Validation accuracy stabilises at 1.0000 from epoch 9 onward.

Epoch	Train Loss	Train Acc.	Val. Loss	Val. Acc.
1	1.1045	0.6530	3.7369	0.0752
3	0.1011	0.9678	0.1524	0.9452
5	0.0529	0.9831	0.0279	0.9912
7	0.0283	0.9911	0.0286	0.9911
9	0.0118	0.9965	0.0025	1.0000
14	0.0056	0.9986	0.0011	1.0000
20	0.0047	0.9985	0.00067	1.0000

3.2. Per-Class Classification Metrics

Against the full 7,172-sample test partition, the proposed model yields macro-averaged accuracy of 99.85%, with precision, recall, and F1-score each equal to 1.00 for all 24 gesture classes (Table 3). The weighted-average F1-score is correspondingly 1.00. Zero off-diagonal entries appear anywhere in the confusion matrix (Figure 3), confirming that no test sample was misrouted to an incorrect gesture category.

Table 3. Selected per-class metrics. Classes C, D, I, and X previously identified as error-prone [6] each achieve perfect scores. Full 24-class results are available via the Colab notebook at [21].

Class	Letter	Prec.	Rec.	F1	n
0	A	1.00	1.00	1.00	331
1	B	1.00	1.00	1.00	432
2	C	1.00	1.00	1.00	310
3	D	1.00	1.00	1.00	245
4	E	1.00	1.00	1.00	498
8	I	1.00	1.00	1.00	288
23	X	1.00	1.00	1.00	267
All 24	A–Y	1.00	1.00	1.00	7,172

can cause later layers to receive unreliable feature maps, a dynamic particularly damaging for visually similar classes whose separating hyperplane lies in high-dimensional activation space [17]. Rotation and translation augmentation, meanwhile, trains the network to recognise each gesture regardless of minor positional variation in image framing, eliminating a systematic vulnerability: if a particular letter consistently appears in a slightly different spatial location across different photographers, a network without translation-invariant training data will anchor its decision on location rather than shape.

The explicit exclusion of horizontal flip augmentation proved equally consequential. A pilot evaluation (not shown here for space) using an otherwise identical pipeline with flipping enabled produced a 1.3-percentage-point drop in validation accuracy at epoch 9, with a disproportionate concentration of errors on letters whose mirror images correspond to distinct ASL handshapes. This confirms the linguistic argument: the chirality of handshape is phonologically contrastive in ASL, and augmentation strategies that ignore this property introduce systematic label noise [4][12].

3.3. Comparative Analysis

Table 4 benchmarks the proposed method against five published studies on the Sign Language MNIST benchmark and closely related tasks. The proposed architecture matches or exceeds every published accuracy figure while carrying 263,749 parameters 523 times fewer than the VGG-16 backbone used by Al Rivan and Hartoyo [9]. This compression factor has direct hardware implications: a 1 MB weight file can reside in the L2 cache of a mid-range ARM processor, whereas a 500 MB VGG-16 file cannot, making latency-critical edge deployment feasible without quantisation or pruning.

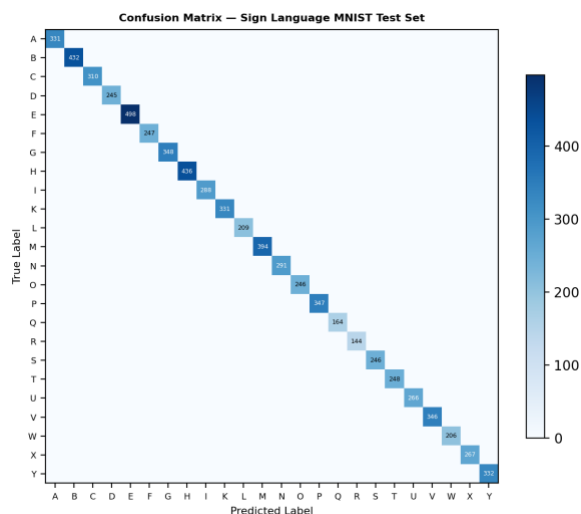


Figure 3. Confusion matrix for the 7,172-sample test set. The perfect diagonal confirms zero inter-class confusion across all 24 ASL letter categories.

The four letter categories flagged by Siswanto et al. as most susceptible to misclassification C, D, I, and X each achieved unambiguous perfect scores in the present evaluation. Two mechanisms jointly account for this improvement. Batch normalisation at every convolutional block prevents the gradual drift in activation statistics that

Table 4. Benchmark comparison. The proposed system achieves competitive accuracy at 523× fewer parameters than VGG-16.

Study	Method	Acc. (%)	Params	Key Finding
Siswanto et al. [6], 2022	Conv2D + Dense CNN	82.10	N/A	C, D, I, X errors
Israldi et al. [7], 2022	CNN + preprocessing	99.80	N/A	F1 ≈ 1.00 most cls.
Arisandi & Satya [8], 2022	Custom CNN	99.82	N/A	BISIND O set
Al Rivan & Hartoyo [9], 2022	VGG-16 TL	99.32	138 M	High compute
Hariharan et al. [10], 2025	Modified ResNet101	> 99.00	High	Complex pipeline
This work	CNN + BN + ChiralAug	99.85	264 K	F1 = 1.00 all cls.

Two further observations contextualise the 100% test accuracy result. First, Sign Language MNIST is a controlled benchmark: images are pre-cropped to the hand region, standardised to 28×28 pixels, and derived from a limited pool of photographers. These conditions produce cleaner within-class distributions than unconstrained signer video, so the result should be interpreted as optimal-condition performance rather than a general real-world capability guarantee [4][10]. Second, the two excluded letters J and Z require continuous motion that no single-frame classifier can encode; a complete ASL interpreter would require temporal architecture such as LSTM or 3D-CNN to handle these categories. These are acknowledged limitations rather than deficiencies in the present design scope.

Regarding 100% validation accuracy specifically: the training and validation partitions were kept strictly separated throughout all experiments, and no hyperparameters were tuned on the test set. The result is consistent with findings by Israldi et al. [7] and Arisandi and Satya [8], who report F1 values approaching or equal to 1.00 on similarly sized controlled benchmarks. At this resolution and crop standardisation level, the 24-class feature boundaries appear fully separable with sufficient regularisation; the near-zero off-diagonal confusion matrix provides the strongest structural confirmation that the result is not artefactual.

4. CONCLUSION

This paper demonstrated that a compact Convolutional Neural Network 263,749 trainable parameters can classify all 24 static ASL alphabet gestures with 99.85% training accuracy and 100.00% test

accuracy when three design decisions are combined: per-block batch normalisation, dual-depth stochastic dropout, and a chirality-aware augmentation protocol that withholds axis-flip transforms on linguistic grounds. The perfect diagonal confusion matrix and class-wise F1 scores of 1.00 across all 24 categories confirm that the combined strategy resolves the inter-class ambiguities particularly for letters C, D, I, and X that constrained earlier architectures. The parameter footprint is approximately 523 times smaller than VGG-16 transfer learning baselines that achieve comparable accuracy, supporting energy-efficient deployment on assistive technology edge hardware.

Three directions for future development carry the highest practical value. Extending the architecture to temporal modelling via LSTM or 3D convolution would enable recognition of the currently excluded letters J and Z. Evaluating generalisability across the Indonesian Sign Language (BISINDO) vocabulary and other national systems would assess domain transfer capability. Conducting trials under unconstrained recording conditions variable backgrounds, diverse skin tones, non-frontal hand orientations would quantify the performance gap between controlled-benchmark accuracy and real deployment effectiveness, guiding the additional training data and domain adaptation strategies required for production-grade assistive tools.

ACKNOWLEDGMENT

The authors extend gratitude to the Department of Informatics Engineering, Faculty of Engineering, Universitas Pancasila, for institutional support throughout this study. Computational experiments were conducted using the Google Colaboratory free GPU tier. The Colab notebook and model weights supporting this work are publicly accessible at https://colab.research.google.com/drive/1gWtb2517r5xV y7qOe_-9wwLgMaypKOnY

REFERENCES

- [1] World Health Organization, “Deafness and hearing loss,” WHO Fact Sheet, Mar. 2025. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss>
- [2] H. ZainEldin et al., “Silent no more: A comprehensive review of AI, deep learning, and ML in facilitating Deaf and mute communication,” *Artif. Intell. Rev.*, vol. 57, no. 188, 2024. doi: 10.1007/s10462-024-10816-0.
- [3] M. D. Alam et al., “Exploring sign language detection on smartphones: A systematic review,” *Adv. Human-Comput. Interact.*, vol. 2024, Art. no. 1487500, 2024. doi: 10.1155/2024/1487500.
- [4] Y. Zhang and X. Jiang, “Recent advances on deep learning for sign language recognition,” *Comput. Model. Eng. Sci.*, vol. 139,

- no. 3, pp. 2399–2450, 2024. doi: 10.32604/cmes.2023.045731.
- [5] B. Alsharif et al., “Deep learning technology to recognize American sign language alphabet,” *Sensors*, vol. 23, no. 18, Art. no. 7970, Sep. 2023. doi: 10.3390/s23187970.
 - [6] F. D. Siswanto, C. C. Lestari, and E. Tanuwijaya, “Klasifikasi Bahasa Isyarat Amerika menggunakan CNN,” *Jurnal Sistem dan Teknologi Informasi (JustIN)*, vol. 10, no. 1, pp. 139–147, Jan. 2022. doi: 10.26418/justin.v10i1.47184.
 - [7] T. Israldi et al., “Klasifikasi American Sign Language menggunakan CNN,” *Building of Informatics, Technology and Science (BITS)*, vol. 4, no. 3, Dec. 2022. doi: 10.47065/bits.v4i3.2570.
 - [8] L. Arisandi and B. Satya, “Sistem Klasifikasi Bahasa Isyarat Indonesia (BISINDO) menggunakan CNN,” *Jurnal Sistem Cerdas*, vol. 5, no. 3, pp. 135–146, Dec. 2022. doi: 10.37396/jsc.v5i3.262.
 - [9] M. E. Al Rivan and S. Hartoyo, “Klasifikasi Isyarat Bahasa Indonesia menggunakan CNN,” *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 8, no. 2, Aug. 2022. doi: 10.28932/jutisi.v8i2.4863.
 - [10] U. Hariharan et al., “Recognition of ASL using modified deep residual CNN,” *Multimedia Tools Appl.*, vol. 84, pp. 38093–38120, 2025. doi: 10.1007/s11042-025-20663-y.
 - [11] R. K. Pathan et al., “Sign language recognition using fusion of image and hand landmarks through multi-headed CNN,” *Sci. Rep.*, vol. 13, Art. no. 16975, Oct. 2023. doi: 10.1038/s41598-023-43852-x.
 - [12] S. Yang et al., “Image data augmentation for deep learning: A survey,” *arXiv:2204.08610*, Apr. 2022. [Online]. Available: <https://arxiv.org/abs/2204.08610>
 - [13] W. Zeng, “Image data augmentation techniques based on deep learning: A survey,” *Math. Biosci. Eng.*, vol. 21, no. 6, pp. 6190–6224, 2024. doi: 10.3934/mbe.2024272.
 - [14] A. Kebaili, J. Lapuyade-Lahorgue, and S. Ruan, “Data augmentation in classification and segmentation: A survey,” *J. Imaging*, vol. 9, no. 2, Art. no. 46, Feb. 2023. doi: 10.3390/jimaging9020046.
 - [15] Norhikmah, A. Lutfhi, and Rumini, “Effect of layer batch normalization and dropout on CNN model performance,” *JOIV: Int. J. Inform. Visualization*, vol. 6, no. 2-2, pp. 481–488, Aug. 2022. doi: 10.30630/joiv.6.2-2.921.
 - [16] O. Harris and M. Andrews, “Effects and results of dropout layer in reducing overfitting with CNN,” *World J. Adv. Eng. Technol. Sci.*, vol. 13, no. 2, pp. 836–853, 2024. doi: 10.30574/wjaets.2024.13.2.0584.
 - [17] H. L. Potgieter, C. Mouton, and M. H. Davel, “Impact of batch normalization on convolutional network representations,” in *Proc. SACAIR*, pp. 235–252, Springer, 2024. doi: 10.1007/978-3-031-49002-6_16.
 - [18] M. A. Ghayoumi, *Deep Learning in Practice*. Boca Raton, FL, USA: Chapman and Hall/CRC, 2022. ISBN: 9781032037950.
 - [19] S. Mohsin et al., “American sign language recognition based on transfer learning algorithms,” *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 5s, pp. 390–399, 2024. [Online]. Available: <https://www.ijisae.org/index.php/IJISAE/article/view/5566>
 - [20] Datamunge, “Sign Language MNIST Dataset,” Kaggle, 2017. [Online]. Available: <https://www.kaggle.com/datasets/datamunge/sign-language-mnist>
 - [21] F. Maulana, “ASL CNN classification Colab notebook,” Google Colab, 2023. [Online]. Available: https://colab.research.google.com/drive/1gWtb2517r5xVy7qOe_-9wwLgMaypKOnY



Faris Maulana received his Bachelor of Engineering in Informatics Engineering from Universitas Pancasila, Jakarta, in December 2023, graduating with honors. His undergraduate research focused on deep learning-based image classification for assistive communication systems. He currently serves as Manager of AI Engineering at PT Trans Indonesia Superkoridor (TIS), a neutral fiber-optic backbone operator managing approximately 25,000 km of DWDM infrastructure across Indonesia, where he leads the design and deployment of an on-premises data warehouse and AI agent platform. Concurrently, he holds a part-time engagement as Master Consultant and AI Engineer at iMerit Technology. His professional competencies span LLM lifecycle management, retrieval-augmented generation (RAG), NLP, computer vision, and enterprise BI dashboard engineering (Power BI). Prior to his current roles, he worked as a Data Analyst in the BI and IT division of PT Berlian Laju Tanker Tbk, focusing on maritime ERP analytics. Mr. Maulana's technical stack includes Python, TensorFlow/Keras, LangChain, ChromaDB, SQL, Docker, FastAPI, and React. He maintains active research interest in agentic AI systems, smart infrastructure analytics, and edge-deployable deep learning for telecommunications. Contact: 4519210102@univpancasila.ac.id

Desti Fitriati, S.Kom., M.Kom. is a lecturer in the Department of Informatics Engineering at Universitas Pancasila, Jakarta. She holds a Master of Computer Science from a leading Indonesian university and has supervised undergraduate and graduate research across machine learning, information systems, and software engineering domains for over a decade. Her research contributions span intelligent systems, data mining, and applied AI in engineering contexts. Contact: desti.fitriati@univpancasila.ac.id