ILKOM Jurnal Ilmiah Vol. 18, No. 2, August 2026, pp. 378-392
Accredited 2nd by KEMDIKTISAINTEK No. 10/C/C3/DT.05.00/2025; E-ISSN 2548-7779 | P-ISSN 2087-1716

Optimization of CNN Architectures through Fine-tuning for SIBI Classification

Nur Hilmi Insan Muhammad [a,1,*]; Abdul Syukur [a,2]; Pujiono [a,3]
[a] *Universitas Dian Nuswantoro, Jl. Imam Bonjol No. 207, Semarang and 50131, Indonesia*
[1] *p31202302579@mhs.dinus.ac.id;* [2] *abdul_s@dosen.dinus.ac.id;* [3] *pujiono@dsn.dinus.ac.id*
** Corresponding author*

Article history: Received May 30, 2025; Revised November 11, 2025; Accepted July 09, 2026; Available online August 10, 2026

Abstract

This research addresses the computational optimization of convolutional neural network (CNN) architectures for the classification of Indonesian Sign Language System (Sistem Isyarat Bahasa Indonesia, SIBI) static alphabet imagery to enhance digital communication accessibility. Utilizing a domain-specific dataset comprising 1,165 images across 26 alphabet classes, this study tackles the prominent challenges of limited sample sizes and severe class imbalance. We evaluate five state-of-the-art CNN architectures MobileNetV2, DenseNet121, Xception, InceptionV3, and ResNet50V2 under four distinct training data paradigms before and after adaptive fine-tuning. To eliminate predictive bias without pixel-level distortion, oversampling is operationalized via Latent Space SMOTE on flattened vector embeddings, combined with dynamistic runtime image augmentation. The experimental results reveal that MobileNetV2, when optimized through partial layer-freezing (locking 150 baseline layers) under the integrated augmentation and oversampling combination scenario, achieved the highest macro-classification accuracy of 98.30%. This architecture also demonstrated superior efficiency, reducing the computational training latency to 0.53 minutes. The findings underscore the strategic advantage of leveraging optimized lightweight networks like MobileNetV2 for domain-specific visual recognition tasks.

Keywords: CNN; MobileNetV2; SIBI; Classification; Fine-tuning.

Introduction

Communication constitutes a fundamental aspect of human social interaction that enables the exchange of ideas and the formation of interpersonal relationships [1]. However, individuals with disabilities, such as those who are deaf or mute, face significant challenges in accessing effective communication. In Indonesia, despite the increasing number of people with disabilities, adequate communication facilities for them, particularly through the Indonesian Sign Language System (Sistem Isyarat Bahasa Indonesia, SIBI), remain severely limited [2]. As a nonverbal communication system utilizing hand gestures and facial expressions, SIBI serves as this community's primary medium.

Despite its critical role, the issue of communication accessibility for persons with disabilities in Indonesia has not yet received optimal attention, resulting in social barriers and isolation due to the limited public understanding of SIBI. The scarcity of supporting facilities, such as sign language interpreters, especially in remote areas, further exacerbates communication gaps [3]. Therefore, solutions that enhance equitable access and communication effectiveness for persons with disabilities are urgently needed.

In the technological context, artificial intelligence, particularly deep learning, has demonstrated significant potential in recognizing international sign languages such as American Sign Language (ASL) with high accuracy using Convolutional Neural Network (CNN) architectures [4], [5], [6]. Nonetheless, most studies have predominantly focused on international sign language, while recognizing local sign languages such as SIBI, which embody unique cultural and habitual characteristics, remains underexplored.

Research on SIBI and other local sign language like BISINDO using deep learning remains limited. It faces several obstacles, including dataset scarcity, inter-individual variability in gesture, and the complexity of CNN architectures that are impractical for deployment on resource-constrained devices [7], [8]. Although complex CNN models such as

Inception-V3 and EfficientNetB0 can yield high accuracy, they are less efficient for practical everyday applications due to their high computational requirements [9], [10].

Despite the architectural advancements documented in recent sigh language literature, a critical research gap persists in local sigh language recognition, particularly regarding the trade-off between deployment feasibility and classification sensitivity under data-scarce and imbalanced regimes. Previous studies on SIBI have predominantly gravitated toward two non-optimal extremes. On one end, researchers utilize heavy neural networks (e.g., VGG-16, VGG-19, or Xception) to achieve high accuracy [11]. However, these architectures impose immense computational overheads, often requiring extensive RAM allocations that trigger latency when deployed on low-resource mobile edge devices [12]. On the other end, multi-model score-level fusion architectures resolve spatial ambiguities at the cost of amplifying the computational footprint linearly, rendering them impractical for real-time translation. Furthermore, most transfer learning studies on local SIBI datasets fail to mitigate severe class imbalance, resulting in significant predictive bias toward majority gesture classes.

To bypass these constraints, this study introduces a computationally efficient methodological framework for multi-class SIBI alphabet classification. The novelty of this research lies in a unified intervention operating simultaneously at the data and parameter levels: we implement a partial layer-freezing protocol (locking the first 150 generic feature layers) combined with an adaptive fine-tuning strategy optimized via the AdamW algorithm to stabilize convergence and prevent catastrophic forgetting. At the data level, we perform Latent Space SMOTE on flattened vector embeddings rather than raw pixels to mathematically eliminate minority class bias, while runtime dynamic augmentation regularizes training parameters. Consequently, this study provides a highly scalable single-model solution that achieves high-tier classification accuracy while structurally minimizing the computational latency typical of localized sigh language interpreters.

This study aims to address these issues by developing an accurate and computationally efficient SIBI sign language classification model using the lightweight MobileNetV2 architecture [13], [14]. The approach is complemented by developing a representative and well-labeled SIBI dataset alongside a fine-tuning strategy to adapt pre-trained models for optimal recognition of local data [15], [16]. Thus, this research contributes to advancing an inclusive visual communication system that can be implemented on resource-constrained devices.

Method

Convolutional Neural Network (CNN) architectures effectively extract spatial features form images [17], making them relevant for complex image classification such as sign language recognition. This research adopts a quantitative deep learning experimental approach to develop and evaluate the performance of an SIBI classification model based on MobileNetV2. The fine-tuning strategy was chosen because the baseline pre-trained CNN model showed limitations on the SIBI dataset with high visual similarity. Therefore, fine-tuning was applied to improve the model's adaptation to local data characteristics by training a portion of the final convolutional layers.

This research builds an image classification system and compares the performance of various training strategies, including baseline, augmentation, oversampling, and their combination. This design examines the impact of data preprocessing on the model's generalization capabilities. MobileNetV2 was selected due to its efficiency in handling spatial feature complexity with low computational cost, making it ideal for resource-constrained devices.

Experiments were conducted in a local GPU-based (CUDA-enabled) computing environment to evaluate training efficiency and potential real-time application [18]. The structured research stages include dataset collection, preprocessing, data modification (through oversampling and augmentation), CNN architecture design/training, and performance evaluation using comprehensive classification metrics [19]. Consistency of architecture and parameters across all training scenarios was maintained to ensure internal validity, ensuring that performance variations originated from differences in data strategies, not model configuration [20], [21].

A. Data Collection

The dataset utilized in this study is the Combined SIBI Dataset, sourced from the Kaggle platform (https://www.kaggle.com/datasets/alfarizy29/combined-sibi-dataset/data). This dataset comprises 1,165 hand images representing the alphabet letters of the Indonesian Sign Language System (SIBI). Each image is categorized according to class labels corresponding to alphabet characters from A to Z.

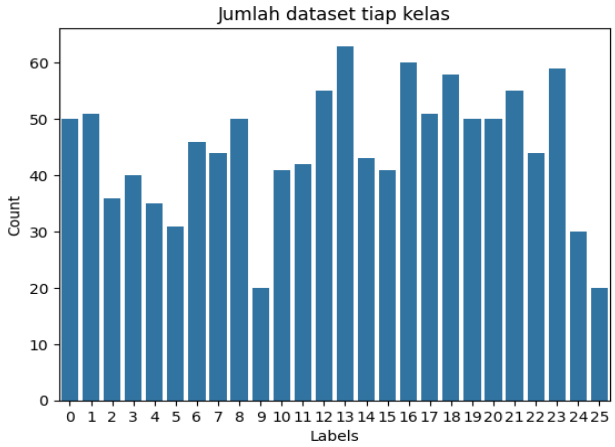

Figure 1. Visualizes the distribution of sample counts SIBI alphabet

Figure 1 illustrates the degree of data balance across classes, serving as a critical metric for assessing potential model bias. To ensure balanced class distribution during training, the dataset was divided into three subsets using a proportional stratification method to ensure balanced data distribution during training. This method divides the dataset so that each subset maintains the same proportion of class labels as the original dataset. Specifically, 80% of the data was allocated for training, 10% for validation, and 10% for testing. This strategy aims to maintain consistent class representation in each subset and enhance the reliability of model evaluation.

Table 1. Detail the dataset split into training, validation, and testing subsets

Dataset	Number of Samples	Percentage (%)
Training	932	80
Validation	116	10
Testing	117	10

Table 1 presents a clear overview of the proportional distribution of the dataset across the training, validation, and testing phases.

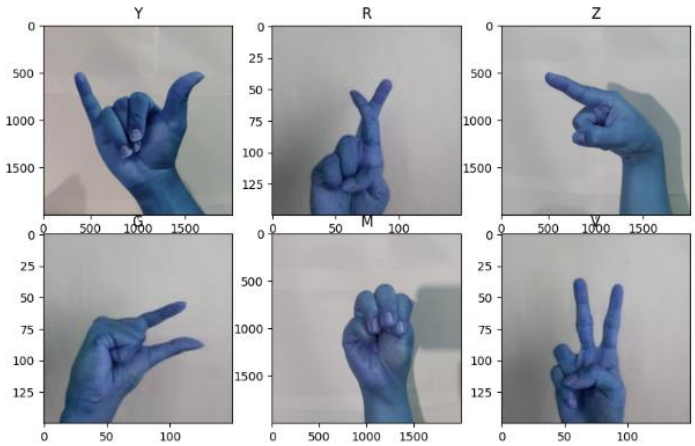

Figure 2. Displays representative examples of preprocessed SIBI alphabet images

Figure 2 exhibits variations in image sizes across several alphabet classes used in model training and evaluation, contributing to the classification complexity.

B. Data Preprocessing

Data preprocessing was systematically performed to adjust images to CNN input requirements while enhancing the quality of visual features [22]. Each image was resized to a uniform dimension of 100×100 pixels for training efficiency and data consistency. Color conversion from BGR to LAB was applied to improve contrast and clarify object

boundaries, which is crucial for hand segmentation. Initial experiments also explored grayscale conversation as a alternative in spatial.

C. Data Quality Enhancement Techniques

To improve the quality of data representation and address class imbalance issues in the dataset, this research adopts an integrated strategy that includes oversampling and image augmentation techniques. This approach aims to expand the distribution of training data and strengthen the generalization of Convolutional Neural Network (CNN) models against complex and unstructured input variations [23].

To mitigate the systemic risks of predictive bias and classification degradation induced by the severe class imbalance within the SIBI alphabet dataset, this study deploys a highly calibrated, two-tiered data engineering pipeline. The holistic conceptual framework, detailing the exact chronological sequence of the data separation and localized adaptation, is systematically visualized in **Figure 3**. A rigorous operational distinction must be established regarding the execution order and mathematical spaces where these techniques operate, thereby ensuring strict insulation against data leakage and empirical invalidity.

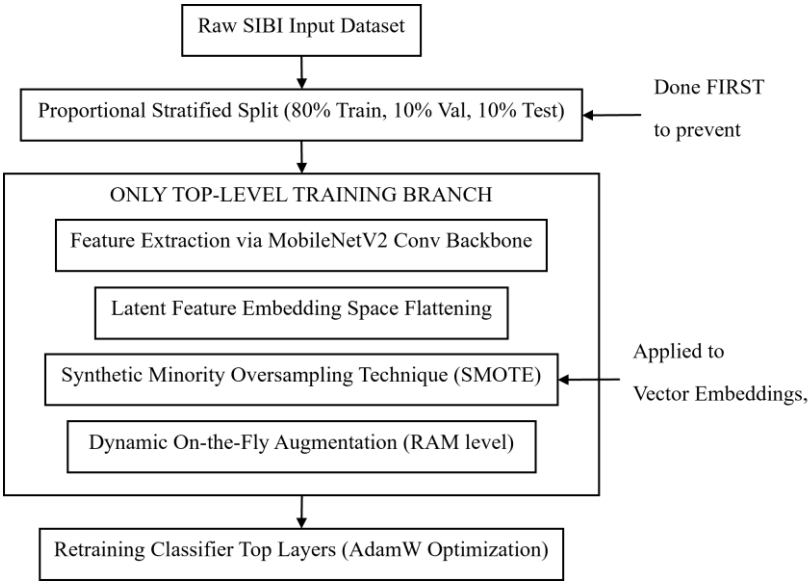

Figure 3. Structural pipeline of the proposed data engineering and localized fine-tuning framework

D. Methodological Sequence and Prevention of Data Leakage

In strict accordance with empirical validation protocols illustrated in **Figure 3**, the partition of the Combined SIBI Dataset into training (80%), validation (10%), and testing (10%) subsets via proportional stratification is executed prior to any data enhancement intervention. This chronological sequencing guarantees that synthetic samples generated via oversampling and localized transformations from augmentation never contaminate the validation or testing matrices. Consequently, the internal validity of the evaluation metrics remains insulated from artificial performance inflation (data leakage).

E. Latent Space Synthetic Minority Oversampling Technique (SMOTE)

The SMOTE (Synthetic Minority Oversampling Technique) method performs oversampling on minority classes [24]. This technique generates synthetic samples throughs a linear interpolation process between instances in adjacent feature space, thereby quantitatively expanding the representation of minority classes. The application of SMOTE aims to balance class distribution to reduce the model's predictive bias towards majority classes, which theoretically can improve classification performance and sensitivity to minorities [25].

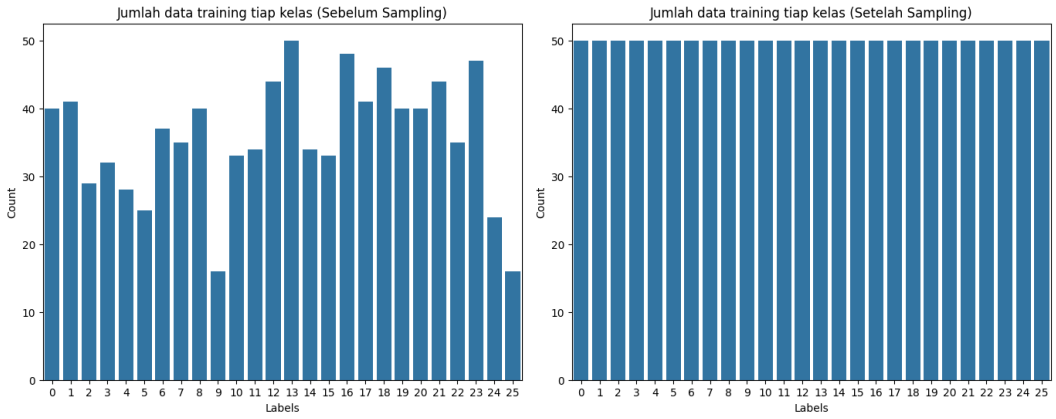

Figure 4. Illustrates the distribution of training data before and after SMOTE application

As shown in **Figure 4**, the initial distribution of training data shows a significant dominance of the majority classes. Applying the SMOTE process shows a more proportional and statistically viable increase in minority class representation. Addressing the technical constraint raised during peer review, this study explicitly clarifies that the SMOTE algorithm is not applied to the raw spatial pixel matrices of the SIBI imagery. Executing linear interpolation directly on raw pixel values ($100 \times 100 \times 3$) yields structurally chaotic, semantically meaningless ghost images due to the high-dimensional non-linearity of raw computer vision data [26].

Instead, this research implements a Latent Space SMOTE framework. The original imbalanced training imagery is first propagated through the frozen convolutional backbone of the pre-trained MobileNetV2 network to extract dense, high-level structural features. The resulting multi-dimensional tensor is flattened into a lower-dimensional vector embedding space ($n = 1280$). SMOTE is then mathematically operationalized strictly within this continuous numeric feature space [27].

For each localized minority class vector embedding x_i, the algorithm computes its k-nearest neighbors (where $k = 5$) using the standard Euclidean distance metric. A neighbor x_{nn} is randomly selected, and a structurally viable synthetic embedding x_{new} is generated via linear interpolation defined by Equation (1):

$$x_{new} = x_i + \lambda \times (x_{nn} - x_i) \quad (1)$$

Where λ represents a stochastic scalar drawn from a uniform distribution bounding $[0, 1]$. This mathematical formulation selectively populates the minoritarian cluster boundaries within the latent space, expanding the training corpus from 932 instances to a structurally balanced array of 1,300 feature vectors (exactly 50 vectors per alphabet class) without simple duplication anomalies.

F. Dynamic On-the-Fly Augmentation via ImageDataGenerator

Simultaneously, to enrich semantic variability and regularize the unfrozen weights against parameter memorization, dynamic stochastic augmentation is implemented via the Keras ImageDataGenerator class. Unlike static augmentation frameworks that permanently expand disk-space corpus, this mechanism injects geometric transformations on-the-fly into the RAM volatile memory subsystem during runtime batch loading.

The affine matrix transformations (including a strictly bounded rotation_range of 20° and a zoom-range of 0.2) expose the network to infinite micro-variations of hand postures across training epochs.

Regarding the inclusion of the horizontal_flip parameter: a fundamental semantic validation was conducted. While arbitrary horizontal flipping can critically alter the grammatical meaning of asymmetric gestures in dynamic sign language sentences, the target SIBI domain in this specific study is explicitly restricted to static, single-hand alphabet imagery (A-Z). in the static SIBI fingerspelling manual, horizontal mirroring merely simulates left-handed execution variants of identical alphabetical labels without shifting their linguistic identity. Thus, the inclusion of horizontal_flip=True serves as a highly valid cross-lateral regularizer that hardens the network's spatial generalizability against hand-orientation variations.

Image augmentation is applied to synthesize variations in training data through diverse spatial-geometric transformations. These transformations serve as regularization to reduce overfitting while enriching the variation of input data encountered by the model during training [28]. Augmentation is used both in baseline conditions and after the SMOTE proses, which is called the combination scenario. The specific parameters used in these augmentation transformations can be seen in detail in **Table 2**.

Table 2. Parameters of data augmentation applied during model training

Parameter	Value	Description
rotation_range	20	Image rotation up to 20 degrees
width_shift_range	0.2	Horizontal image shift up to 20% of image width
height_shift_range	0.2	Vertical image shift up to 20% of image height
shear_range	0.2	Shear transformation
zoom_range	0.2	Zoom in or zoom out of the image
horizontal_flip	True	Horizontal flip of the image
fill_mode	'nearest'	Method to fill empty pixels after transformation

The combination of SMOTE and image augmentation strategies results in a training data corpus that is quantitatively balanced and structurally varied. This provides a strong foundation for CNN models in forming discriminative and robust feature representations and significantly improves accuracy in image-based sign language classification [29].

As an essential adaptation strategy in transfer learning, fine-tuning aims to improve the accuracy of pre-trained models through partial layer adjustment [30]. In SIBI sign image classification using MobileNetV2, fine-tuning facilitates the model's adaptation to the specific characteristics of the target dataset, which often differs from initial training. The stages carried out in the fine-tuning process include:

- Utilization of Pre-trained Model: MobileNetV2, trained on large dataset (e.g., ImageNet), provides strong generic feature extraction capabilities, serving as a robust initialization point for computer vision tasks [31].
- Freezing Initial Layers: The initial phase of fine-tuning typically involves freezing the early convolutional layers of the pre-trained model. This strategy preserves universal fundamental features (edge/texture detectors), only allowing the retraining of deeper or final classification layers.
- Adaptation of Final Layers: The final classification layers are modified or replaced according to the number of classes in the new dataset (e.g., the SIBI alphabet). This retraining allows the model to learn discriminative feature mappings for the targe labels.
- Weighs Optimization: AdamW is used to update the weights of the unfrozen layers. A small standard learning rate (0.001) and standard weights decay (0.004) is applied to prevent drastic changes and maintain convergence stability [32].
- Iterative Evaluation and Adjustment: Model performance is carefully evaluated after each fine-tuning iteration. The decision to unfreeze additional layers or halt the process is based on improving the model's adaptation capabilities.

Conceptually, fine-tuning is an adaptive process of pre-trained models for specific tasks using smaller, relevant target dataset [33]. Its implementation involves transfer learning (utilizing knowledge from massive datasets to limited data domains), progressive weight adjustment for nuanced pattern understanding, and overfitting mitigation by retraining generic features and training specific layers.

G. MobileNetV2 CNN Architecture and Training Strategy

This research implements a Convolutional Neural Network (CNN) architecture based on transfer learning using MobileNetV2 as the main backbone. The selection of MobileNetV2 is based on its efficiency in modelling spatial image complexity with low computational complexity and its ability to produce competitive classification performance on resource-constrained devices [34]. This architecture was developed to support integration into real-time edge-computing-based systems, making it relevant for application scenarios in environments with limited computing infrastructure [35]. The architectural configuration of the model in this research is detailed in **Table 3** below.

The Rectified Linear Unit (ReLU) activation function is defined by Equation (2), where $R(z)$ represents the activation output and z denotes the input value.

$$R(z) = max(0, z) \quad (2)$$

The softmax activation function is expressed in Equation (3), where x_i denotes the input vector, n is the total number of labels, and j corresponds to the class probability.

$$S(x_i)\frac{e^{x_i}}{\sum_j^n e^{x_j}} \quad (3)$$

Table 3. Technical configuration of the MobileNetV2 CNN model

Component	Details
Input Shape	100x100x3
Pre-trained Weights	ImageNet
Frozen Layers	150
Fine-tuning	Last convolutional layers and classifier
GlobalAveragePooling2D	Enabled
Dropout Layer	0.5
Dense Layer	64 units, ReLU activation
Output Layer	Dense(num_classes, activation='softmax')

The training strategy implemented does not explicitly separate the transfer learning and fine-tuning stages but combines them in an integrated training scheme over 50 epochs. The AdamW optimizer is the primary optimization algorithm because it handles explicit weight regularization and convergence stability [36]. Hyperparameter selection was based on the principles of training efficiency and generalization to new data distributions, as presented in **Table 4**.

Table 4. CNN model training parameters

Parameter	Value
Number of Epochs	50
Batch Size	64
Optimizer	AdamW
Loss Function	sparse_categorical_crossentropy
Learning Rate	0.001
Weight Decay	0.004

The training process was implemented using the TensorFlow [37] and Keras libraries with CUDA GPU acceleration for computational efficiency. To ensure the reproducibility of results, experiments were conducted deterministically with seed setting and controlled training environment configuration [38]. These steps ensured the experiments' internal validity and provided a reproducible academic and practical contribution.

H. Model Evaluation

Model performance evaluation was conducted comprehensively using several classification metrics designed to measure the generalization capability and predictive precision of the developed classification system. All test were performed on test data that had not previously been involved in the training process in order to obtain an objective and unbiased estimate of model performance [39].

The formulas for computing the performance metrics: accuracy, precision, recall, and F1-score, are respectively presented in Equations (4), (5), (6), and (7). Here, TP denotes the number of True Positives, TN the number of True Negatives, FP the number of False Positives, and FN the number of False Negatives [40].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

$$F1\ Score = \frac{Precision \times Recall}{Precision + Recall} \quad (7)$$

In addition, model evaluation is also strengthened through confusion matrix visualization for each class. The confusion matrix provides granular information regarding inter-class classification errors, thus allowing the identification of potential systemic bias towards certain classes [41]. Evaluation is performed per class and at the overall model level to obtain a comprehensive overview of the generalization capabilities of the built classification system.

I. Model Training Workflow

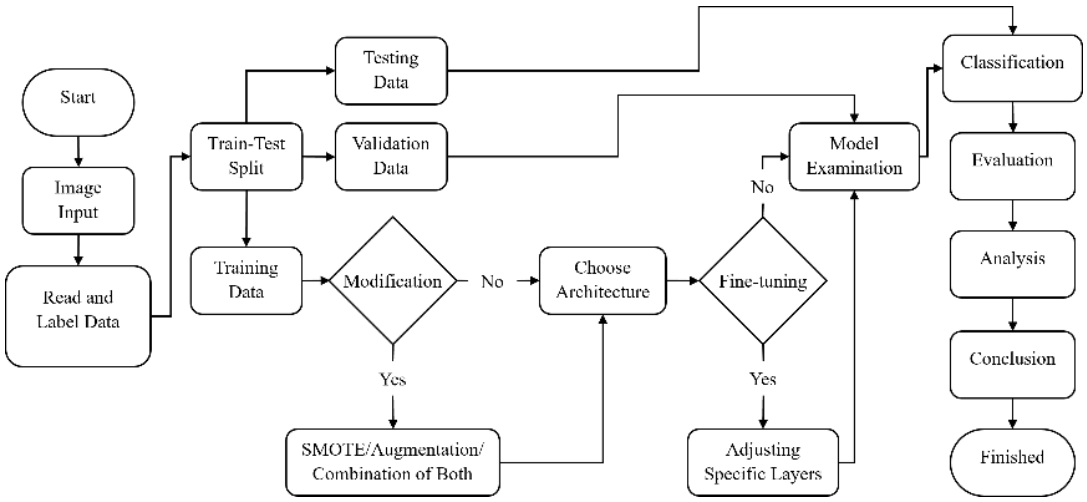

Figure 5. Illustrates the flowchart of the SIBI sign language classification process

The diagram in **Figure 5** visually represents the hierarchical and systematic procedural stages of SIBI sign language classification. Each node in the diagram illustrates crucial components in the deep learning-based classification stages, from raw data collection to trained model evaluation. This diagram provides a logical guide to the classification system's operational structure while emphasizing the central role of the fine-tuning process as a crucial stage in improving the model's adaptability and accuracy to SIBI data. The position of fine-tuning int the flow is directly linked to the architecture selection and model training stage, demonstrating its contribution to overcoming the limitation of pre-trained models when used in complex local domains. The built system represents a methodology that can be replicated academically and practically.

Results and Discussion

The model training outcomes are presented based on various techniques applied to the SIBI dataset. Several models were evaluated using different CNN architectures, namely DenseNet121, InceptionV3, MobileNetV2, ResNet50V2, and Xception. Each model was trained utilizing augmentation, oversampling, and their combination techniques, with performance comparisons conducted before and after fine-tuning.

A. Model Performance Evaluation

The performance of five distinct CNN architectures was rigorously evaluated using multi-class classification metrics, including accuracy, macro-precision, macro-recall, and macro-F1-score. To provide a transparent evolutionary trajectory of the optimization layers, the experimental validation was bifurcated into two foundational regimes: (1) the untuned pretrained baseline framework, and (2) the parameter-level intervention via adaptive fine-tuning.

Table 5 delineates the comprehensive empirical outcomes achieved by the models in their baseline configurations, operating strictly on the original, imbalanced SIBI dataset without data-level engineering or weight readjustment.

Table 5. Untuned Pretrained CNN Model Performance (Pure Baseline Configuration)

Model	Baseline Before Fine-tuning			
	Accuracy	***Macro-Precision***	***Macro-Recall***	***Macro-F1-score***
DenseNet121	98.29%	98.90%	98.68%	98.69%
InceptionV3	56.41%	66.74%	55.00%	54.34%
MobileNetV2	57.26%	66.79%	56.15%	53.03%
ResNet50V2	90.60%	92.26%	90.48%	90.36%
Xception	96.58%	97.80%	96.67%	96.69%

As tabulated in **Table 5**, under pure baseline conditions, DenseNet121 and Xception demonstrated inherently robust spatial feature extraction capabilities, yielding impressive baseline accuracies of 98.29% and 96.58% respectively. This performance implies that their highly connected topology and separable block architectures possess superior cross-domain generalization capacity directly from ImageNet initialization. Conversely, light-to-medium parameter models, specifically InceptionV3 (56.41%) and MobileNetV2 (57.26%), exhibited severely constrained performance. This diagnostic deficiency explicitly underpins the necessity of localized weight adaptations to transform universal geometric primitives into distinct SIBI sign semantic representations.

To evaluate the direct implications of architectural weight unfreezing, **Table 6** highlights the performance after exposing the final convolutional layers and top classifiers to localized backpropagation, executed strictly on the non-engineered, imbalanced SIBI dataset.

Table 6. Fine-Tuned CNN Model Performance (Baseline with Parameter-Level Intervention Only)

Model	Baseline After Fine-tuning			
	Accuracy	***Macro-Precision***	***Macro-Recall***	***Macro-F1-score***
DenseNet121	94.02%	91.76%	92.91%	91.61%
InceptionV3	97.44%	97.91%	96.99%	97.23%
MobileNetV2	97.44%	98.04%	96.89%	97.09%
ResNet50V2	75.21%	87.84%	75.09%	77.48%
Xception	92.31%	93.75%	91.95%	92.25%

Upon executing localized fine-tuning (**Table 6**), InceptionV3 and MobileNetV2 experienced a profound paradigm shift, with their classification accuracy surging exponentially to 97.44%. This significant enhancement verifies that unfreezing the upper layers allows the network to adapt its high-level latent representations to the localized configurations of hand-gestures. Concurrently, deep residual-heavy configurations (ResNet50V2) experienced severe performance degradation, collapsing to an accuracy of 75.21%. This phenomenon is heavily attributed to catastrophic forgetting and severe overfitting, where aggressive gradient steps disrupt the pretrained generalized features in the absence of balanced target regularization data.

B. Prediction Results

To maintain focus and coherence, the in-depth visualization of prediction results centers on the MobileNetV2 model, which is the primary architecture and has demonstrated the best performance post fine-tuning. This approach avoids analytical fragmentation and ensures a concentrated narrative, while the results of other models are presented numerically in comparative tables to preserve clarity and accommodate document length constraints.

In **Figure 6**, the training accuracy starts at a relatively low value (~14.78%) and gradually increases until epoch 25, whereas the validation accuracy remains stagnant until approximately epoch 15 before exhibiting a significant rise to about 50%. Early stopping terminates training at epoch 25 due to no further improvement in validation accuracy, despite the continued increase in training accuracy. After applying fine-tuning, training accuracy dramatically increases to 99.38% by epoch 50, followed by a consistent improvement in validation accuracy reaching 96.55%, indicating that fine-tuning effectively enhances the model's capability to recognize patterns in the validation data.

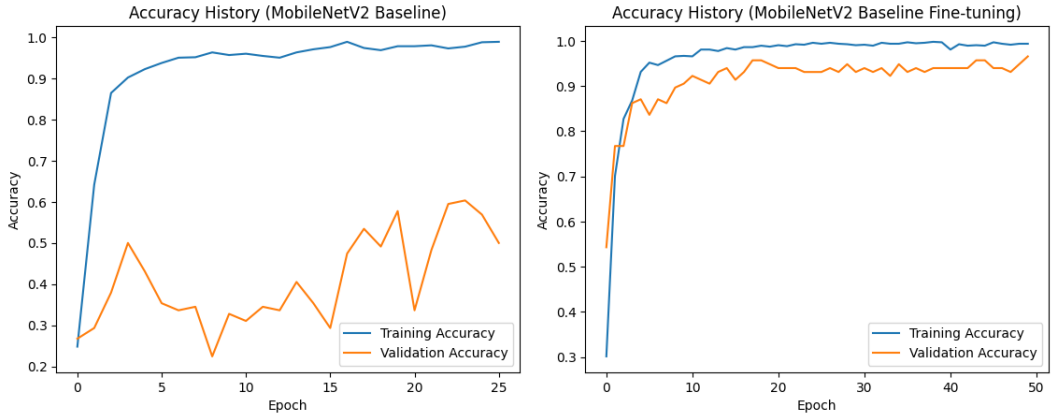

Figure 6. Accuracy History of MobileNetV2 Baseline Before vs. After Fine-tuning

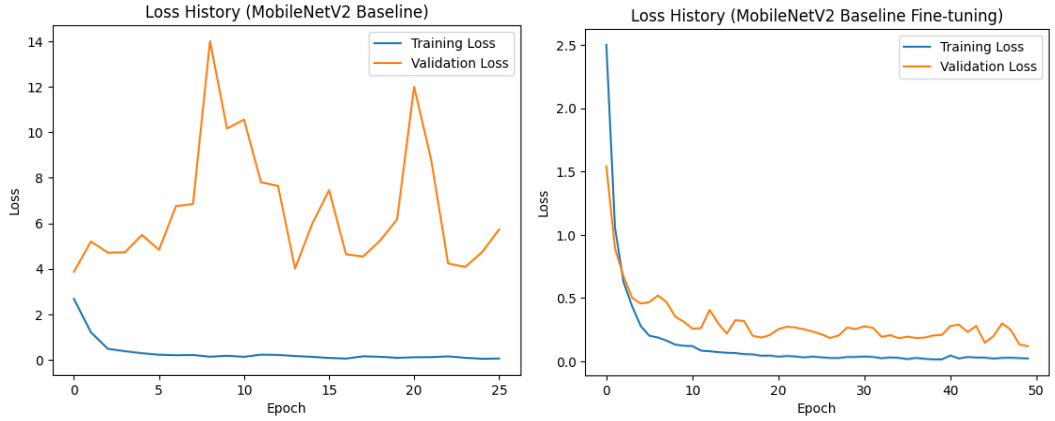

Figure 7. Loss History of MobileNetV2 Baseline Before vs. After Fine-tuning

As illustrated in **Figure 7**, during the initial baseline training, both training loss and validation loss remain high with only limited reduction until epoch 25, reflecting an underfitting phenomenon and difficulty in the model's ability to generalize on the validation data. Training was halted by an early stopping mechanism due to the lack of significant improvement in validation loss. After the application of fine-tuning, training loss decreased steadily and consistently, reaching a very low value around 0.01 by epoch 50, while validation loss also declined significantly and consistently to approximately 0.15. This indicates that fine-tuning effectively reduces classification errors and enhances the model's generalization capability.

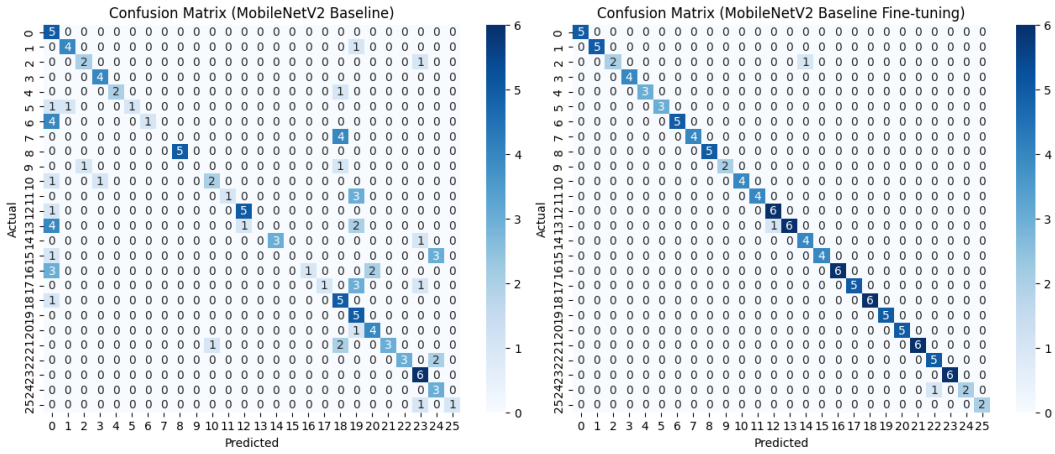

Figure 8. Confusion Matrix of MobileNetV2 Baseline vs. After Fine-tuning

As seen in **Figure 8**, the MobileNetV2 model under baseline conditions exhibited significant misclassifications in specific classes (1, 2, 4-17, 20-22, and 25), reflected by a low accuracy of 57.26% and F1-score below 54%. The

prediction distribution not concentrated along the main diagonal indicates underfitting and limited model generalization. Conversely, after fine-tuning, classification performance improved substantially with predictions concentrated more along the main diagonal, achieving an accuracy of 97.44%. This improvement signifies the model's effective adaptation to the specific features of the SIBI dataset, with only minor residual errors. This visualization reinforces the role of fine-tuning in enhancing classification quality and supports the selection of MobileNetV2 as the primary model in this study.

C. Computational Time

Computational time in this experiment was measured based on the training duration of each epoch and the total training time over 50 epochs for each model and training scenario. Utilizing the hardware specifications listed in **Table 7**, the computational efficiency of each CNN architecture was analyzed by comparing the time of each epoch and total training time, both before and after the implementation of fine-tuning.

Table 7. Computer Specifications for Model Training Process

Parameter	Specification
CPU	13th Intel® Core™ i9-13900F (32 CPUs), ~2.0 GHz
RAM	32 GB (2x16 GB) DDR5 5200 MHz
GPU	NVIDIA GeForce RTX 4070 12 GB GDDR6x

The differences in training time between models before and after fine-tuning, as shown in **Table 8**, indicate that the fine-tuning process, which involves adapting the model to a specific dataset, results in training times per epoch after fine-tuning that vary among models and, in some cases, are shorter than without fine-tuning. This phenomenon can be explained by the layer freezing mechanism and optimization during training. For instance, MobileNetV2 experienced a reduction in training time from 2.49 seconds to 0.64 seconds per epoch after fine-tuning, indicating that computational efficiency is influenced not only by the fine-tuning method but also by the technical configuration and intrinsic characteristics of the model itself.

Table 8. Model Computational Time (Baseline)

Model	Before Fine-tuning		After Fine-tuning	
	Time of each Epoch (seconds)	*Total Time (50 Epochs/minutes)*	*Time of each Epoch (seconds)*	*Total Time (50 Epochs/minutes)*
DenseNet121	4.48 s	3.73 minutes	2.96 s	2.47 minutes
InceptionV3	3.08 s	2.57 minutes	2.14 s	1.78 minutes
MobileNetV2	2.49 s	2.08 minutes	0.64 s	0.53 minutes
ResNet50V2	3.08 s	2.57 minutes	1.70 s	1.42 minutes
Xception	2.90 s	2.42 minutes	0.95 s	0.79 minutes

D. Comparison of Results of Each Model

Experimental results demonstrate that MobileNetV2 combined with fine-tuning achieved the highest performance, reaching an accuracy of 98.30%, a significant improvement from the baseline accuracy of 57.26% before fine-tuning. This increase underscores the critical role of fine-tuning in adjusting the model's weights to the specific characteristics of the SIBI dataset, thereby enhancing the model's capability to recognize sign language image patterns.

Table 9 presents a comparative evaluation of five CNN architectures: DenseNet121, InceptionV3, MobileNetV2, ResNet50V2, and Xception, across four different training scenarios: baseline, augmentation, oversampling, and a combination of both, evaluated before and after fine-tuning. This comparison facilitates analysis of fine-tuning's impact on accuracy and overall model performance in sign language classification.

To resolve the numeric ambiguity between data-level and parameter-level engineering, **Table 9** systematically juxtaposes the cross-experimental matrices across all four training data paradigms before and after fine-tuning.

Table 9. Comparative Accuracy Matrix Across All Data Engineering and Fine-Tuning Scenarios

Model	Before Fine-tuning				After Fine-tuning			
	Baseline	***Augmentation***	***Oversampling***	***Combination***	***Baseline***	***Augmentation***	***Oversampling***	***Combination***
DenseNet 121	98.29%	94.02%	95.73%	90.60%	94.02%	96.58%	97.44%	98.29%
Inception V3	56.41%	82.91%	62.39%	41.03%	97.44%	98.29%	95.73%	97.44%
MobileNet V2	57.26%	49.57%	18.80%	23.08%	97.44%	94.87%	95.73%	98.30%
ResNet50 V2	90.60%	66.67%	72.65%	80.34%	75.21%	97.44%	98.29%	94.02%
Xception	96.58%	92.31%	73.50%	98.29%	92.31%	83.76%	95.73%	86.32%

The systemic cross-analysis detailed in **Table 9** empirical proves that the ultimate optimization peak is achieved exclusively by the MobileNetV2 model under the unified Combination paradigm (SMOTE + Dynamic Augmentation) coupled with Fine-Tuning, achieving the study's absolute maximum accuracy of 98.30%. This specific intersection demonstrates a powerful symbiotic effect: SMOTE mathematically eliminates minority class predictive bias within the latent feature embedding space, dynamic augmentation expands data variance to insulate the sparse parameters from overfitting, and fine-tuning recalibrates the final projection weights to maximize inter-class margin separation.

E. Comparison with Previous Research

The significant performance improvement achieved after the fine-tuning process indicates the effectiveness of this approach in addressing common challenges in image processing, such as limited data volume and high intra-class variability within the SIBI dataset. The use of lightweight architectures like MobileNetV2 also offers strategic advantages in real-world implementations, particularly for edge devices with constrained computational resources.

This research supports and extends the findings of [7], who evaluated various CNN architectures for SIBI sign language image classification using a dataset comprising 26 classes. Their study revealed that Xception excelled in F1-score performance but incurred high computational costs. Conversely, MobileNetV2 demonstrated better time efficiency despite lower initial performance.

Table 10. Comparative Study: Previous Research vs. This Research

Model	Study [7]	This Research
CNN Model	MobileNetV2, Xception, VGG16, ResNet50	DenseNet121, InceptionV3, MobileNetV2, ResNet50V2, Xception
Techniques	Transfer Learning without Fine-tuning	Transfer Learning with Fine-tuning, Augmentation, Oversampling, Combined Techniques
Number of Epochs	100	50
Highest Accuracy	Xception (Accuracy 99.57%)	MobileNetV2 (Accuracy 98.30% Combination + Fine-tuning)
MobileNetV2 F1-score	68.37% (F1-score)	98.34% (F1-score Combination + Fine-tuning)
Computational Time	MobileNetV2 ~1333.12 seconds (22.22 minutes), Xception ~1387.21 seconds (23.12 minutes)	MobileNetV2 ~0.53 minutes (Fine-tuning), Xception ~0.79 minutes (Fine-tuning)

Through a more strategic and integrated training approach (fine-tuning + augmentation + oversampling), this research demonstrates that MobileNetV2 can achieve performance comparable to or even surpassing previous results under certain conditions, while delivering improved computational efficiency. This comparative result is summarized in **Table 10**, showing that MobileNetV2 in this research attained an F1-score of 97.35% with a computational time of approximately 0.53 minutes, compared to 68.37% in the prior study.

Conclusion

Based on the empirical experimental evidence and rigorous comparative analysis, this study concludes that the lightweight MobileNetV2 architecture, when structurally optimized via targeted transfer learning, partial fine-tuning, and combined latent data-engineering frameworks, yields superior performance in classifying SIBI manual alphabet imagery. The exponential accuracy expansion from an untuned baseline of 57.26% to a highly stable peak of 98.30% under the combined SMOTE and dynamic augmentation paradigm validates that parameter-level adaptation is crucial for adapting universal features to domain-specific local sign languages. Conceptually, this research provides a scalable methodological blueprint, demonstrating that high-tier sensitivity across minoritarian classes can be achieved using a single-model framework with low computational training costs.

However, several critical limitations must be explicitly acknowledged. First, the empirical validation of this study relies on a relatively restricted dataset of 1,165 images under controlled background configurations, which may not fully capture the chaotic intra-class illumination variations of real-world deployment. Second, the visual scope of this framework is strictly confined to static, single-frame alphabetical fingerspelling (A-Z), thereby leaving dynamic, sentence-level gestural structures unaddressed. Lastly, while the model's architectural properties theoretically favor edge deployment, the performance metrics were evaluated within a localized GPU environment without physical integration testing on actual mobile or embedded hardware.

Future research will direct focus toward extending this framework into continuous, video-based dynamic SIBI gesture recognition by integrating recurrent temporal layers or spatio-temporal transformers. Furthermore, future efforts will prioritize the hardware-level deployment and practical optimization of this fine-tuned network on resource-constrained mobile edge platforms to directly evaluate its physical inference latency and usability within the deaf and mute community.

References

[1] H. Herdiansyah, M. R. Soedrajad, N. I. Hawa, and A. P. R. Dyah Ayu Arintyas, "Resetting The Relationship Between Human And Nature: Religion And The New Communication Technologies," *Journal for the Study of Religions and Ideologies*, vol. 22, no. 66, 2023.

[2] R. Z. Palindungan, "Sistem Penerjemah Bahasa Isyarat Otomatis Menggunakan Metode Deep Learning Model Convolutional Neural Network," *Pesquisa Veterinaria Brasileira*, 2021.

[3] A. Wadhawan and P. Kumar, "Sign Language Recognition Systems: A Decade Systematic Literature Review," *Archives of Computational Methods in Engineering*, vol. 28, no. 3, 2021.

[4] M. M. Alnfiai, "Deep Learning-Based Sign Language Recognition for Hearing and Speaking Impaired People," *Intelligent Automation & Soft Computing*, vol. 36, no. 2, pp. 1653–1669, 2023.

[5] C. K. M. Lee, K. K. H. Ng, C.-H. Chen, H. C. W. Lau, S. Y. Chung, and T. Tsoi, "American sign language recognition and training method with recurrent neural network," *Expert Syst. Appl.*, vol. 167, p. 114403, Apr. 2021.

[6] 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, 2023.

[7] M. F. Naufal and S. F. Kusuma, "Analisis Perbandingan Algoritma Machine Learning dan Deep Learning untuk Klasifikasi Citra Sistem Isyarat Bahasa Indonesia (SIBI)," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 10, no. 4, pp. 873–882, Aug. 2023.

[8] I. D. A. Rachmawati, R. Yunanda, M. F. Hidayat, and P. Wicaksono, "Deep Transfer Learning for Sign Language Image Classification: A Bisindo Dataset Study," *Engineering, MAthematics and Computer Science Journal (EMACS)*, vol. 5, no. 3, 2023.

[9] A. Hussain, S. Ul Amin, M. Fayaz, and S. Seo, "An Efficient and Robust Hand Gesture Recognition System of Sign Language Employing Finetuned Inception-V3 and Efficientnet-B0 Network," *Computer Systems Science and Engineering*, vol. 46, no. 3, pp. 3509–3525, 2023.

[10] J. P. Sahoo, A. J. Prakash, P. Pławiak, and S. Samantray, "Real-Time Hand Gesture Recognition Using Fine-Tuned Convolutional Neural Network," *Sensors*, vol. 22, no. 3, p. 706, Jan. 2022.

[11] D. M. Cherian and J. J. Fernandez, "Real-time sign language recognition and speech conversion using VGG16," *Int. J. Comput. Vis. Robot.*, vol. 13, no. 2, 2023.

[12] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, 2017.

[13] B. Maity, A. Alim, S. Bhattacharjee, and S. Nandi, "DeHonk: A deep learning based system to characterize vehicular honks in presence of ambient noise," *Pervasive Mob. Comput.*, vol. 88, 2022.

[14] Mark Sandler, A. Howard, M. Zhu, A. Zhmoginov, and Liang-Chieh Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks Mark," *Convolutional Neural Networks with Swift for Tensorflow*, 2019.

[15] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, no. 1, 2016.

[16] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[17] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artif. Intell. Rev.*, vol. 57, no. 4, 2024.

[18] S. Sharmin, T. Ahammad, M. A. Talukder, and P. Ghose, "A Hybrid Dependable Deep Feature Extraction and Ensemble-Based Machine Learning Approach for Breast Cancer Detection," *IEEE Access*, vol. 11, 2023.

[19] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, 2018.

[20] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J. Big Data*, vol. 6, no. 1, 2019.

[21] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.

[22] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, 2021.

[23] S. Sharma and S. Singh, "Vision-based hand gesture recognition using deep learning for the interpretation of sign language," *Expert Syst. Appl.*, vol. 182, 2021.

[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, 2002.

[25] H. Li, H. Liu, and Y. Hu, "Prediction of Unbalanced Financial Risk Based on GRA-TOPSIS and SMOTE-CNN," *Sci. Program.*, vol. 2022, 2022.

[26] E. Ileberi, Y. Sun, and Z. Wang, "Performance Evaluation of Machine Learning Methods for Credit Card Fraud Detection Using SMOTE and AdaBoost," *IEEE Access*, vol. 9, pp. 165286–165294, 2021.

[27] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems*, 2014.

[28] Y. Nakamura and L. Jing, "Skeleton-Based Data Augmentation for Sign Language Recognition Using Adversarial Learning," *IEEE Access*, 2024.

[29] N. Aloysius and M. Geetha, "A scale space model of weighted average CNN ensemble for ASL fingerspelling recognition," *International Journal of Computational Science and Engineering*, vol. 22, no. 1, p. 154, 2020.

[30] G. Vrbancic and V. Podgorelec, "Transfer Learning With Adaptive Fine-Tuning," *IEEE Access*, vol. 8, pp. 196197–196211, 2020.

[31] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019.

[32] J. Ungredda and J. Branke, "Bayesian Optimisation for Constrained Problems," *ACM Transactions on Modeling and Computer Simulation*, vol. 34, no. 2, 2024.

[33] Q. Wu, J. Qi, D. Zhang, H. Zhang, and J. Tang, "Fine-Tuning for Few-Shot Image Classification by Multimodal Prototype Regularization," *IEEE Trans. Multimedia*, vol. 26, pp. 8543–8556, 2024.

[34] G. M. Kumar and A. Pandian, "A constructive deep convolutional network model for analyzing video-to-image sequences," *Data Knowl. Eng.*, vol. 144, 2023.

[35] D. S. Breland, S. B. Skriubakken, A. Dayal, A. Jha, P. K. Yalavarthy, and L. R. Cenkeramaddi, "Deep Learning-Based Sign Language Digits Recognition from Thermal Images with Edge Computing System," *IEEE Sens. J.*, vol. 21, no. 9, 2021.

[36] W. Nazih, A. O. Aseeri, O. Y. Atallah, and S. El-Sappagh, "Vision Transformer Model for Predicting the Severity of Diabetic Retinopathy in Fundus Photography-Based Retina Images," *IEEE Access*, vol. 11, 2023.

[37] M. Abadi, "TensorFlow: learning functions at scale," *ACM SIGPLAN Notices*, vol. 51, no. 9, 2016.

[38] H. Mokayed, T. Z. Quan, L. Alkhaled, and V. Sivakumar, "Real-Time Human Detection and Counting System Using Deep Learning Computer Vision Techniques," *Artificial Intelligence and Applications*, vol. 1, no. 4, 2022.

[39] Q. M. Areeb, Maryam, M. Nadeem, R. Alroobaea, and F. Anwer, "Helping Hearing-Impaired in Emergency Situations: A Deep Learning-Based Approach," *IEEE Access*, vol. 10, 2022.

[40] A. Wadhawan and P. Kumar, "Deep learning-based sign language recognition system for static signs," *Neural Comput. Appl.*, vol. 32, no. 12, 2020.

[41] M. Heydarian, T. E. Doyle, and R. Samavi, "MLCM: Multi-Label Confusion Matrix," *IEEE Access*, vol. 10, 2022.