**Comparative Analysis of Machine Learning and Deep Learning Models for Hand Sign Language Classification: Performance Evaluation and Comparison**

*Kevin* Chunady1, *Felicia* Natania2, *Simeon* Yuda Prasetyo3

1,2,3Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia

**Abstract.** The presence of computer vision has great application potential to improve accessibility for communication for the deaf community and the larger community that tends not familiar with sign language. In this paper, we present a comparative analysis of several machine learning and deep learning approaches for sign language recognition using image classification. The machine learning approach uses models such as Decision Tree, K-Nearest Neighbors (KNN), Stochastic Gradient Descent (SGD), Support Vector Classifier (SVC), and Logistic Regression while the deep learning approach uses CNN models which are DenseNet121, MobileNetV2, NasNetMobile, ResNet50, VGG19, and a custom architecture. This paper also utilizes the hyperparameter tuning method to enhance the machine learning models and the fine-tuning method to enhance the deep learning models. The ASL dataset is pre-processed and evaluated through accuracy, precision, recall, and F1-score. The result of this comparative study indicates that generally, the deep learning approaches outperform the basic machine learning algorithms in terms of accuracy and other performance metrics with an overall of above 90% accuracy.

**Keyword.** Sign Language Classification, Machine Learning, Convolutional Neural Networks, Transfer Learning, Fine-tuning

**1 Introduction**

Communication is crucial in everyday life, it is the process of exchanging information which is part of the human interaction needed in each society. Sign language greatly facilitates communication in the global hearing-impaired community. In 2021, 430 million individuals, or more than 5% of the world's population suffer from hearing loss [1]. These restrictions encourage the usage of sign language or non-verbal language to help people with disabilities communicate. Sign language varies in different countries, for instance, American Sign Language (ASL) is the standard in America. Despite these limitations, very few people understand sign language as it is not mandatory to learn them. This complicates the communication process between the deaf community and the hearing majority. With these complications, it is essential to develop a system for deciphering sign language as this breaks the communication barrier between vocal-hearing disabled people and a common person.

With the technology that continues to advance in this era, machine learning technology may be used to design systems for the identification of sign language since it is so good at identifying objects or classifying data. In this research, convolutional neural network, which is a subset of machine learning, is used as the basis of the sign language recognition system. An architecture is needed for a recognition system using a convolutional neural network. With the given condition, DenseNet121, MobileNetV2, NasNetMobile, ResNet50, VGG19, and a custom architecture are used as this architecture. This research uses six architectures for deep learning so that these architectures can be analyzed on which architecture is better to be used on the sign language recognition system. These architectures are typically divided into two main parts [2]. The first step is feature extraction, which involves taking the desired features out of a picture using computer vision or image processing methods. The second component, the recognizer, should be able to correctly identify the testing data on which the algorithms were used and learn the pattern from the extracted and defined characteristics.

On the other hand, this research implements 5 different machine learning models which are Decision Tree, K-Nearest Neighbors (KNN), Stochastic Gradient Descent (SGD), Support Vector Classifier (SVC), and Logistic Regression. This paper compares Deep Learning and Machine Learning approaches in terms of detecting Sign Language Detection.

For the past three decades, researchers have worked to address problems in sign language detection, especially in the field of machine learning. Over the years the development of computer vision has been enhanced deeply, with different techniques. The goal of these research is to create functional translation systems that can translate sign language to spoken language and vice versa. Various studies have pointed to the implementation of deep learning procedures to undertake Sign Language Recognition.

In 2021, a study aimed to optimize the recognition system of ASL sign language by applying preprocessing to the system and then comparing the recognition accuracy [3]. This paper uses a series of combinations in the preprocessing process including grayscale, HSV colorspace and background elimination, global thresholding, and adaptive thresholding using the convolutional neural network architecture of MobileNetV2. Using this technique, they obtained grayscale, HSV colorspace, background elimination, and global thresholding scenarios had the best improvement combined with a recognition accuracy of 97%, which indicates the improvement in classification accuracy.

Similar research has been done in 2022 [4]. This paper also made ASL sign language recognition using CNN. The difference between this research and the research before [3] is that this research preprocesses images only by resizing them to 64 x 64 x 3 pixels. Also, standard MobileNet is used instead of MobileNetV2. With this model, an accuracy of 95,41% to 20 epochs was achieved. This indicates the proposed model in this paper can effectively predict and classify different sign languages. But, applying more preprocessing and using MobileNetV2 achieved higher accuracy which can be seen in the previous paper [3].

In another research which was conducted in 2021, a recognition system for Nigerian Sign Language was made [5]. This research uses MobileNetV2 as the main CNN architecture. With a dataset consisting of around 5000 images including 137 sign words, preprocessing was done by resizing them to 224 by 224 pixels. After 60 epochs, the model obtained 43.97% of accuracy. This indicated that the model needs fine-tuning. In 154 layers of MobileNetV2, the first 100 layers were left frozen. The model obtained a test accuracy of 91.15% after training for an additional 140 epochs. This shows fine-tuning can greatly enhance the classification system made with CNN.

A study conducted in 2020 was done with an improved AlexNet for image classification tasks [6]. In this paper, it was explained many parameters that are connected can cause problems with traditional CNN. To conquer this, a new method was by adding a deconvolution layer. The accuracy improved by 3%. This shows that CNN models can be modified to make the performance better.

Another study was made in 2020 on welding defect detection [7]. In this study, 5 deep learning architectures which were MobileNet, Xception, VGG16, VGG19, and ResNet50 were used. Basic machine learning methods were also used in this study. For MobileNet, a modified version is made which is the TL-MobileNet model. In this model, DropBlock was set to 0.8 and it is proven that the best recognition accuracy is obtained with a smaller model size. This study shows that multiple deep learning and basic machine learning models can be used for classification tasks and convolution optimization techniques can increase model accuracy.

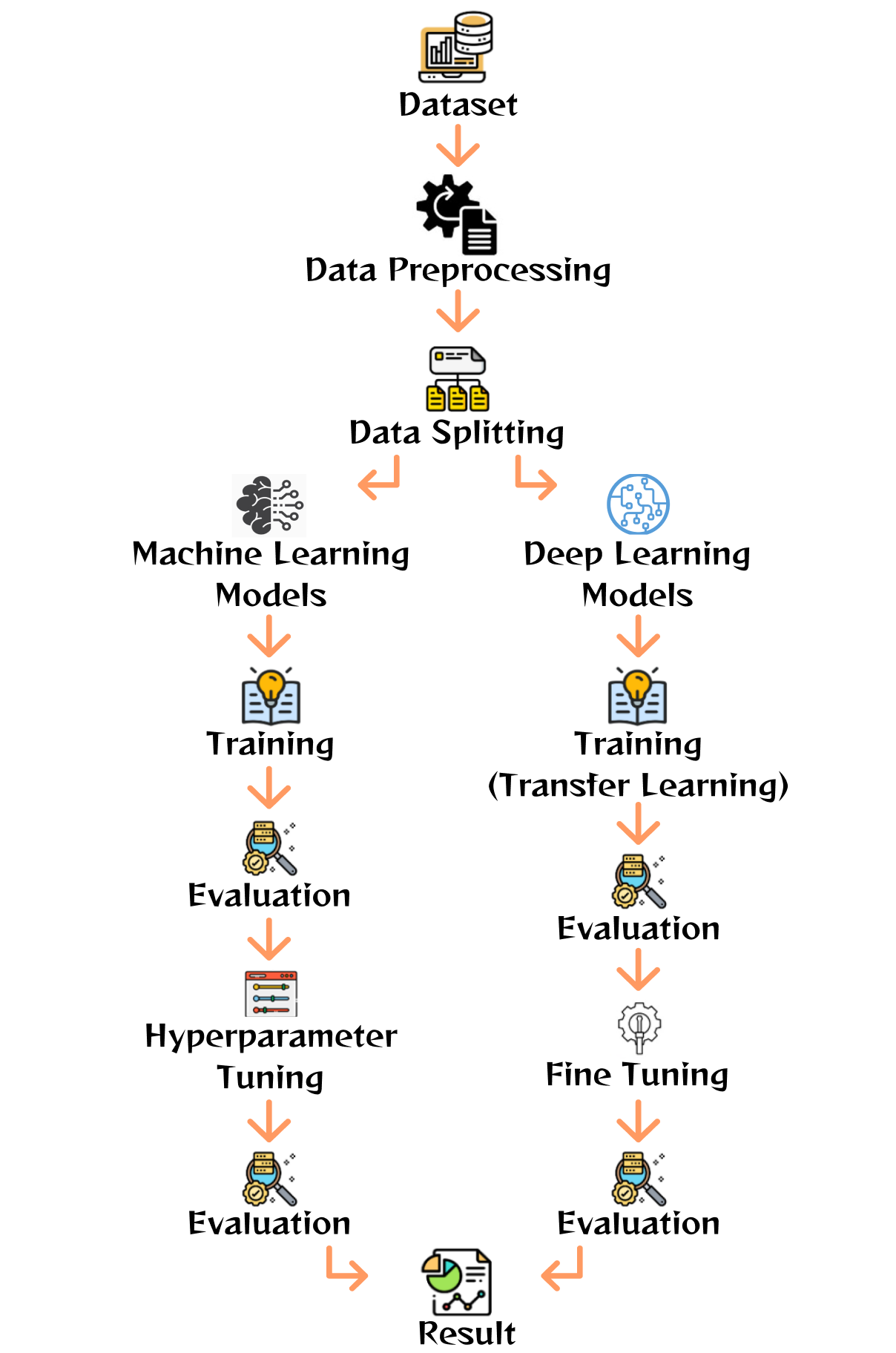
Another research in 2021, also applied a deep learning and machine learning approach [8]. This paper uses the SVM, KNN, Random Forest, Decision Tree, Naive Bayes, ANN, and MLP algorithms to obtain the average accuracy. The MediaPipe framework is used in this paper for its feature extraction method to be used in many datasets. Among them are ASL (Alphabet), Indian (Alphabet), Italian (Alphabet), ASL (Numbers), and Turkey (Alphabet). Results show that SVM outperformed the other machine learning algorithms. Additionally, SVM also achieved higher accuracy than deep learning algorithms. The SVM model received an average accuracy of 99% in most of the sign language datasets with the help of MediaPipe’s technology. This concludes that deep learning models aren't always better than machine learning models.

This paper's objective is to compare both deep learning models and machine learning algorithms in sign language recognition using image classification. This research aims to provide insights for a suitable approach by evaluating different algorithms’ performance based on their accuracy and computational efficiency. Using a dataset consisting of American Sign Language images, this research trains and tests the dataset with several machine learning and deep learning algorithms. Furthermore, performed fine-tuning and hyperparameter tuning to the algorithms. To evaluate the performance, this research used accuracy, precision, recall, and F1-score.

The remainder of this paper is organized as follows: in Chapter 2, the research method is described. In Chapter 3, the performed experimental results are discussed. Finally, the conclusions are drawn in Chapter 4.

**2 Methodology**

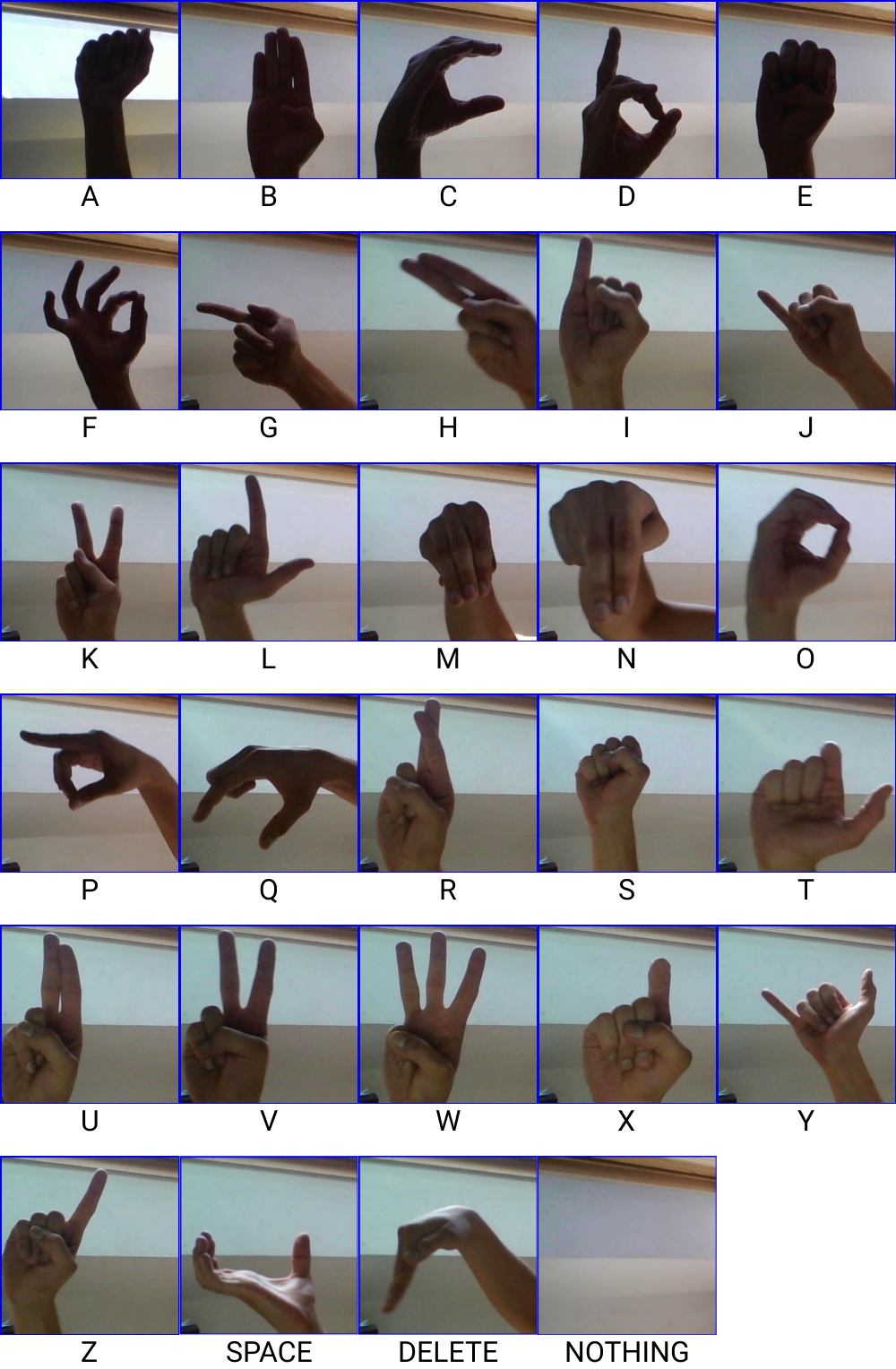
The workflow of the research method is presented in Figure 1. Detailed information about this workflow is explained in this chapter.



**Fig. 1.** Sign Language Classification Workflow

**2.1 Dataset**

The proposed sign language recognition system uses the dataset obtained from Kaggle with the title ASL Alphabet which is made by Akash [9]. The dataset contains pictures of alphabet sign language from the American Sign Language (ASL). This dataset contains 87.000 images which are in the format of jpg and size 200 x 200 pixels. It consists of 29 classes, 26 of the classes are the letters A to Z and there are 3 classes for SPACE, DELETE, and NOTHING. Each class contains exactly 3.000 images. Examples of the images in each class are presented in Figure 2.



**Fig. 2.** Dataset Visualization

Then, using a ratio of 80:10:10, these 87,000 photos are split into a training set, a validation set, and a test set. Thus, the training set consists of 69.000 photos, the validation set of 8.700 images, and the test set of 8.700 images.

**2.2 Preprocessing**

The collection of alphabet sign language images used in this study was originally composed of pixel values. To simplify the data so that it can be easily recognized by deep learning and machine learning models, the photos are downsized from 200 x 200 pixels to 32 x 32 pixels. This data is then normalized. Each pixel is labeled at a number between 0-255. To normalize this, dividing each pixel number by 255 is necessary with the purpose of each pixel value being between 0 and 1. With this data, each class already has quite enough images of 3.000 images with a total of 72.000 images. This shows class balance and sufficiency meaning that data augmentation is not needed. Data is now ready to be trained using deep learning and machine learning models.

**2.3 Basic Machine Learning**

For this comparative analysis, several basic machine learning algorithms are included such as Decision Tree, K-Nearest Neighbors (KNN), Stochastic Gradient Descent (SGD), Support Vector Classifier (SVC), and Logistic Regression.

Decision Tree is a supervised learning approach that may be applied to classification. The algorithm divides the feature space into a tree-like structure, each leaf node represents a class label and each inside node represents a feature. The class label of a given image can then be established using a hierarchical structure of decision rules that are created using the decision tree technique [10]. K-Nearest Neighbors (KNN) is a non-parametric method, the algorithm assigns a label or value to a query point based on the majority judgment or mean of the labels or values of the query point's k nearest neighbors in the training data. Although the distance metric used to evaluate the similarity between points may change depending on the data and task, Euclidean distance is a prominent option [11]. Stochastic Gradient Descent (SGD) is a commonly used optimization algorithm that is used for training machine learning models, the algorithm categorizes photos based on their pixel values or features retrieved from the images, SGD can be used for image classification [12], [13].

Support Vector Classifier (SVC), a supervised learning method used for classification tasks, using a nonlinear model this algorithm can handle large dimensional data [14]. Logistic Regression is a supervised learning approach used for classification tasks. The method operates by analyzing the data and fitting a probability function, which converts the input features into a likelihood score between 0 and 1 [15].

For each basic machine learning algorithm, a grid search approach is applied to optimize the hyperparameters, such as the n-neighbors and kernel type, to achieve the best performance.

**2.4 Deep Learning**

Deep learning is a branch of machine learning that uses multiple-layered neural networks to learn data representations like the human brain [16]. In this paper, several pre-trained deep learning algorithms which are DenseNet121, MobileNetV2, NasNetMobile, ResNet50, and VGG19 are used for sign language recognition using image classification. In addition, it also makes use of a customized Convolutional Neural Network (CNN) model.

The DenseNet121 model consists of a total of 121 layers, of which are convolutional layers, pooling layers, and a final classification layer [17]. The MobileNetV2 introduces a new type of residual connection called the inverted residual block, which further improves the efficiency and performance of the network [18] The NasNetMobile model is based on a stack of blocks, each block contains several convolutional layers and a reduction layer [19]. The ResNet50 model is made of 50 layers [20]. The VGG19 model is composed of 19 layers which include 16 convolutional layers and 3 fully connected layers [21].

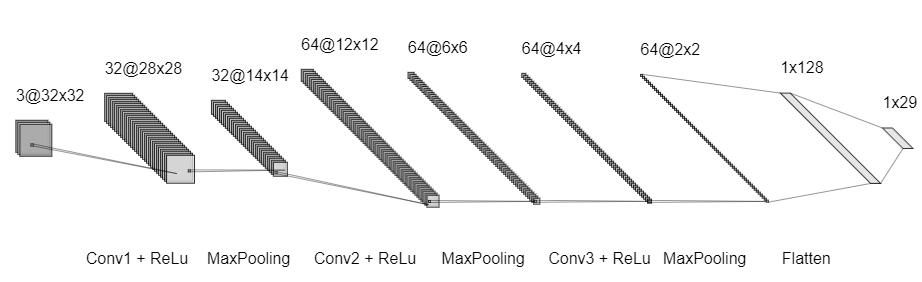
All pre-trained deep learning models are used in the same method of transfer learning and fine-tuning. The first thing done is creating the base model from the models pre-loaded with weights trained on the ImageNet dataset, a big dataset with one thousand classes and one million photos. The top classification layers of this base model are not put into the network, which is perfect for feature extraction. Also, the input shape is adjusted according to the data image size which is 32x32x3. The next step is to freeze the convolutional base created. This prevents the weights in the layers of the base model from being updated during training. Then, on top of the base model, we build our classifiers. In particular, we utilize a global average pooling layer, a dense layer using relu activation function with 1.000 classifiers as an output, and a dense layer using softmax activation function with 29 classifiers as an output. The final output for each model would contain 29 classes to perform classification.

After all models are made, we now compile the models before training them. With a learning rate of 0,0001 and category cross-entropy loss, we employ adam optimizer for compiling. Training the models can now be done. We train the models for 10 epochs and evaluate their performance by the metrics used.

The next method is the fine-tuning process where we also train the weights of the layers in the pre-trained models. The weights need to be adjusted as a result of the training phase to switch from general feature maps to features unique to the dataset. To do this, unfreezing the base model from all the models is done.

The last step is to recompile the models and resume training them. With a learning rate of 0,00001 and category cross-entropy loss, we recompile using the adam optimizer. Because we are training a much bigger model and wish to readjust the pre-trained weight, we must use a lower learning rate than before. For training, we train the models for another 10 epochs and reevaluate their performance by the same metrics as before.

Aside from pre-trained CNN models, a custom CNN model is also used. The architecture for this custom CNN is illustrated in Figure 35.



**Fig. 3.** Custom CNN Architecture.

The CNN model used in our investigation has several layers, as shown in Figure 35. The first layer serves as the input layer for 32x32x3-pixel pictures. There are three convolutional layers (Conv1, Conv2, and Conv3) in the feature extraction section. Convolution filters have dimensions of 5x5, 3x3, and 3x3 for ConvNet1, ConvNet2, and ConvNet3, respectively. 32 filters make up ConvNet1, 64 filters make up ConvNet 2, and 64 filters make up ConvNet 3. Rectified linear units (ReLu) come after each convolution process. MaxPooling is used with 2x2 dimensions after ReLu. The purpose of pooling is to avoid losing crucial data while the feature is represented. Flattening is used for the classification stage after the convolutional stage. Fully linked layers are used in the classification step, which is then followed by a ReLu activation function and a single SoftMax output layer. We studied this model without transfer learning and fine-tuning. What we do is compile and train the model. The adam optimizer and category cross-entropy loss are used to compile the model. Training the model is done for 10 epochs. Finally, we evaluate the model by the metrics used.

**2.5 Evaluation Metrics**

Common performance measures used to assess the effectiveness of classification models in machine learning include accuracy, precision, recall, and F1-score [22].

The percentage of examples in a dataset that were properly categorized is known as accuracy. It is determined as:

|  |  |
| --- | --- |
| Accuracy = | (1) |

The percentage of events accurately categorized as positive out of all instances projected to be positive is known as precision. It is determined as:

|  |  |
| --- | --- |
| Precision = | (2) |

Recall quantifies the percentage of all positively identified cases in the dataset that were correctly categorized. It is determined as:

|  |  |
| --- | --- |
| Recall = | (3) |

When the dataset is unbalanced, the F1-score, which is the harmonic mean of accuracy and recall, is a useful statistic. It is determined as:

|  |  |
| --- | --- |
| F1-Score = | (4) |

**3 Result and Discussion**

All calculations in this paper were done on a laptop with AMD Ryzen 7 4800H with Radeon Graphics, 2.9 GHz, and 16 GB RAM, using Google Colab with the Python programming language.

**Table 1.** Average Accuracy Comparison Table for Basic Machine Learning Algorithm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1 Score |
| Decision Tree | 0.8842 | 0.8800 | 0.8800 | 0.8800 |
| Decision Tree (Hyperparameter Tuning) | 0.6160 | 0.6700 | 0.6200 | 0.6200 |
| K-Nearest Neighbours (KNN) | 0.9764 | 0.9800 | 0.9800 | 0.9800 |
| K-Nearest Neighbours (KNN) (Hyperparameter Tuning) | 0.9948 | 0.9900 | 0.9900 | 0.9900 |
| Stochastic Gradient Descent (SGD) | 0.7019 | 0.7800 | 0.7000 | 0.7100 |
| Stochastic Gradient Descent (SGD) (Hyperparameter Tuning) | 0.6015 | 0.7200 | 0.6000 | 0.6000 |
| Support Vector Classifier (SVC) | 0.9284 | 0.9300 | 0.9300 | 0.9300 |
| Support Vector Classifier (SVC) (Hyperparameter Tuning) | 0.7628 | 0.7700 | 0.7600 | 0.7600 |
| Logistic Regression | 0.5553 | 0.5600 | 0.5600 | 0.5500 |
| Logistic Regression (Hyperparameter Tuning) | 0.5492 | 0.5500 | 0.5500 | 0.5500 |

For the basic machine learning approach, we compared several basic machine learning algorithms as seen in Table 1. The algorithms evaluated in this paper were Decision Tree, K-Nearest Neighbors (KNN), Stochastic Gradient Descent (SGD), Support Vector Classifier (SVC), and Logistic Regression. The database was trained and tested with each algorithm, then the performance of each algorithm was evaluated based on accuracy, precision, recall, and F1 Score. It is shown that the KNN received the highest accuracy of 97.64%, followed by SVC with an accuracy of 92.84%. Following this, the Decision Tree and SGD algorithms achieved moderate accuracies of 88.42% and 70.19%, respectively, while Logistic Regression had the lowest accuracy of 55.53%.

After evaluating the results of the basic machine learning algorithms, hyperparameter tuning using GridSearchCV was applied to each of the algorithms to analyze if the accuracy improves. However, only KNN achieved better accuracy after the hyperparameter tuning process which is 99.48%. No other models became better. This might occur because the tuned customized settings selected are less desirable than the default ones. This can be solved by using different optimization techniques such as RandomizedSearch.

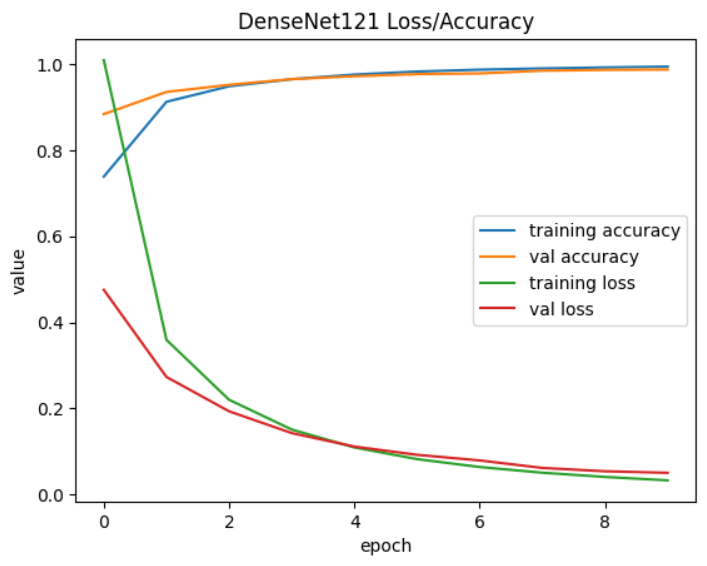
KNN, SVC, and Decision Tree are the three top models after hyperparameter tuning. Parameters after optimization for KNN are n\_neighbors = 3, weights = 'distance', p = 1, while for SVC are 'C' = 0.1, 'gamma' = 10, 'kernel': 'poly', and finally for Decision Tree 'criterion': 'entropy', 'max\_depth' = 9, 'max\_features' = 'sqrt', 'min\_samples\_leaf' = 2, 'min\_samples\_split' = 7.

Compared to previous research presented in the introduction, our research has only the KNN model that can compete with their models. Previous research can achieve above 85% accuracy for basic machine learning models. This can be caused by not using MediaPipe’s technology [8].

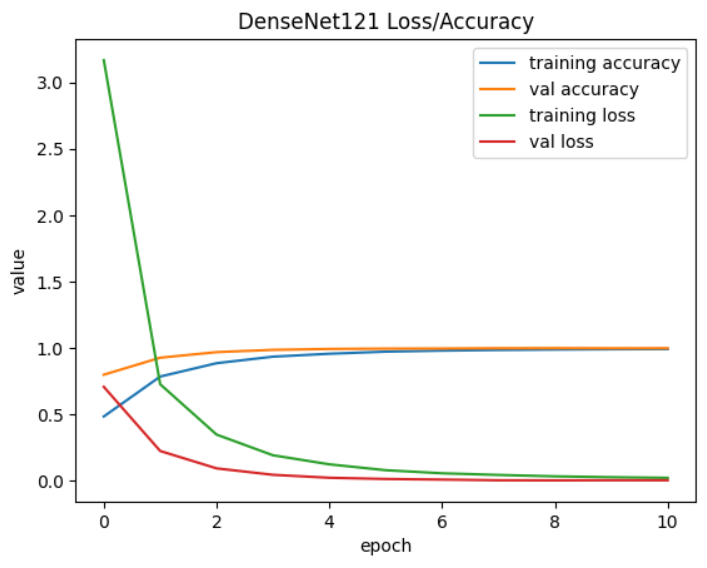
**Table 2.** Average Accuracy Comparison Table for Deep Learning Algorithm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1 Score |
| DenseNet121 | 0.9849 | 0.9853 | 0.9851 | 0.9851 |
| DenseNet121 (Fine-tuned) | 0.9989 | 0.9989 | 0.9989 | 0.9989 |
| MobileNetV2 | 0.4915 | 0.5168 | 0.4908 | 0.4948 |
| MobileNetV2 (Fine-tuned) | 0.9056 | 0.9049 | 0.9049 | 0.9044 |
| NasNetMobile | 0.8833 | 0.8838 | 0.8828 | 0.8825 |
| NasNetMobile (Fine-tuned) | 0.9338 | 0.9361 | 0.9353 | 0.9325 |
| ResNet50 | 0.4864 | 0.5890 | 0.4877 | 0.4810 |
| ResNet50 (Fine-tuned) | 0.8553 | 0.8836 | 0.8564 | 0.8632 |
| VGG19 | 0.9406 | 0.9418 | 0.9411 | 0.9407 |
| VGG19 (Fine-tuned) | 0.9936 | 0.9937 | 0.9938 | 0.9937 |
| CNN | 0.9892 | 0.9900 | 0.9900 | 0.9900 |

Based on the results presented in Table 2 which shows deep learning algorithm results, it can be seen clearly that fine-tuning a deep learning model can lead to significant improvements in its performance. This is proven as every deep learning model which is fine-tuned has better accuracy, precision, recall, and f1 score than before they were fine-tuned. Before fine-tuning, the best performing model was DenseNet121 with an accuracy score of 98.49%, followed by VGG19 with an accuracy score of 94.06%, NasNetMobile with an accuracy score of 88.33%, MobileNetV2 with an accuracy score of 49,15%, and ResNet50 with an accuracy score of 48.64%. However, after fine-tuning, the DenseNet121 model achieved the best performance with an accuracy score of 99.89%, followed by VGG19 with an accuracy score of 99.36%, NasNetMobile with an accuracy score of 93.38%, MobileNetV2 with an accuracy score of 90.56%, and ResNet50 with an accuracy score of 85.53%. MobileNetV2 and ResNet50 show huge improvements in performance during fine-tuning with more than 40% improvement in accuracy score for MobileNetV2 and more than 35% improvement in accuracy score for ResNet50. CNN with custom architecture on the other hand achieved an accuracy score of 98.92%. This shows that this custom architecture is a reliable model with a good accuracy score but not better than fine-tuned VGG19 and DenseNet121.



**Fig. 4.** Accuracy and Loss Graph of DenseNet121.



**Fig. 5.** Accuracy and Loss Graph of Fine-tuned DenseNet121.

Figures 4 and 5 show the best results of transfer learning and fine-tuning, which is the DenseNet121 model. Fine-tuning a model resulted in a significant increase in the performance, in this case, the accuracy of the DenseNet121 model increased from 0.9849 to 0.9989 and the loss of the DenseNet121 model decreased from 0.0514 to 0.0040. From the graphs, we can see that the model has a good fit, not underfitting and overfitting. This can be seen by the training accuracy convergent with the validation accuracy and the training loss convergent with the validation loss.

The result of deep learning models were great and have results comparable with the research done explained in the introduction. From previous research, deep learning models could be further improved by more trial and error such as trying different hyperparameters, models, number of layers that are fine-tuned, and others. Nevertheless, by observing the current performance metrics, the models could already be implemented in a real-life case, which is sign language classification.

**4 Conclusion**

This paper evaluated several deep learning and basic machine learning algorithms for sign language recognition using the American Sign Language (ASL) dataset for comparative study. That showed the deep learning algorithms, specifically DenseNet 121, MobileNet v2, NasNetMobile, ResNet50, and VGG19, outperformed the basic machine learning algorithms, achieving accuracies above 90%. However, it also showed that the basic machine learning algorithms, such as Decision Tree, K-Nearest Neighbors (KNN), Stochastic Gradient Descent (SGD), Support Vector Classifier (SVC), and Logistic Regression, can still achieve reasonable accuracy on the ASL dataset. Overall, deep learning algorithms are highly effective for sign language recognition, especially when large datasets and high computational resources are available, especially after they are fine-tuned. However, basic machine learning algorithms can still be a viable option for sign language recognition when the dataset size and computational resources are limited.

Future research can concentrate on creating more advanced deep learning models for optimizing sign language recognition such as ensemble voting. Additionally, we plan to explore more advanced machine learning algorithms and feature extraction techniques and other advanced fine-tuning techniques to improve the accuracy of sign language recognition. In conclusion, the detection of sign language is a critical implementation of image classification, and both deep learning and basic machine learning algorithms have shown promising results. When choosing a method, one should take into account the size and complexity of the dataset as well as the available computer resources.

**References**

[1] WHO, “Deafness and hearing loss. In World Health Organization.,” Feb. 27, 2023.

[2] Y. H. Liu, “Feature Extraction and Image Recognition with Convolutional Neural Networks,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Oct. 2018. doi: 10.1088/1742-6596/1087/6/062032.

[3] Iduar, I. P., Gede, I. K., Putra, D., & Arya, I. P. Classification of Sign Language Numbers Using the CNN Method. JITTER- Jurnal Ilmiah Teknologi Dan Komputer. (2021)

[4] T. N. Abu-Jamie, P. Samy, and S. Abu-Naser, “Classification of Sign-Language Using MobileNet-Deep Learning,” 2022. [Online]. Available: www.ijeais.org/ijaisr

[5] S. Kolawole, O. Osakuade, N. Saxena, and B. K. Olorisade, “Sign-to-Speech Model for Sign Language Understanding: A Case Study of Nigerian Sign Language,” Nov. 2021, doi: 10.24963/ijcai.2022/855.

[6] S. Li, L. Wang, J. Li, and Y. Yao, “Image Classification Algorithm Based on Improved AlexNet,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Feb. 2021. doi: 10.1088/1742-6596/1813/1/012051.

[7] H. Pan, Z. Pang, Y. Wang, Y. Wang, and L. Chen, “A New Image Recognition and Classification Method Combining Transfer Learning Algorithm and MobileNet Model for Welding Defects,” *IEEE Access*, vol. 8, pp. 119951–119960, 2020, doi: 10.1109/ACCESS.2020.3005450.

[8] A. Halder and A. Tayade, “Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning,” 2021. [Online]. Available: www.ijrpr.com

[9] Akash, “ASL Alphabet,” 2019.

[10] B. Charbuty and A. Abdulazeez, “Classification Based on Decision Tree Algorithm for Machine Learning,” *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, Mar. 2021, doi: 10.38094/jastt20165.

[11] L. Farokhah and P. Korespondensi, “IMPLEMENTASI K-NEAREST NEIGHBOR UNTUK KLASIFIKASI BUNGA DENGAN EKSTRAKSI FITUR WARNA RGB IMPLEMENTATION OF K-NEAREST NEIGHBOR FOR FLOWER CLASSIFICATION WITH EXTRACTION OF RGB COLOR FEATURES”, doi: 10.25126/jtiik.202072608. (2020)

[12] P. Zhou, J. Feng, C. Ma, C. Xiong, and S. Hoi, “Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning.” (2020)

[13] F. Chollet, *Deep Learning with Python*. Manning Publications, 2021.

[14] B. H. Cho, Y. H. Kim, K. B. Lee, Y. K. Hong, and K. C. Kim, “Potential of Snapshot-Type Hyperspectral Imagery Using Support Vector Classifier for the Classification of Tomatoes Maturity,” *Sensors*, vol. 22, no. 12, Jun. 2022, doi: 10.3390/s22124378.

[15] Nazish, S. I. Ullah, A. Salam, W. Ullah, and M. Imad, “COVID-19 Lung Image Classification Based on Logistic Regression and Support Vector Machine,” in *Lecture Notes in Networks and Systems*, Springer Science and Business Media Deutschland GmbH, 2021, pp. 13–23. doi: 10.1007/978-3-030-77246-8\_2.

[16] L. Li, “Application of deep learning in image recognition,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Dec. 2020. doi: 10.1088/1742-6596/1693/1/012128.

[17] X. Li, X. Shen, Y. Zhou, X. Wang, and T. Q. Li, “Classification of breast cancer histopathological images using interleaved DenseNet with SENet (IDSNet),” *PLoS One*, vol. 15, no. 5, May 2020, doi: 10.1371/journal.pone.0232127.

[18] R. Indraswari, R. Rokhana, and W. Herulambang, “Melanoma image classification based on MobileNetV2 network,” in *Procedia Computer Science*, Elsevier B.V., 2021, pp. 198–207. doi: 10.1016/j.procs.2021.12.132.

[19] F. Martínez, F. Martínez, and E. Jacinto, “Performance Evaluation of the NASNet Convolutional Network in the Automatic Identification of COVID-19,” vol. 10, no. 2, 2020.

[20] D. Faye, I. Diop, N. Mbaye, and D. Dione, “Global Journal of Computer Science and Technology Interdisciplinary A Combination of Data Augmentation Techniques for Mango Leaf Diseases Classification A Combination of Data Augmentation Techniques for Mango Leaf Diseases Classification,” 2023, [Online]. Available: https://data.mendeley.com/datasets/hxsnvwty3r

[21] M. Bansal, M. Kumar, M. Sachdeva, and A. Mittal, “Transfer learning for image classification using VGG19: Caltech-101 image data set,” *J Ambient Intell Humaniz Comput*, Apr. 2021, doi: 10.1007/s12652-021-03488-z.

[22] A. Tasnim, Md. Saiduzzaman, M. A. Rahman, J. Akhter, and A. S. Md. M. Rahaman, “Performance Evaluation of Multiple Classifiers for Predicting Fake News,” *Journal of Computer and Communications*, vol. 10, no. 09, pp. 1–21, 2022, doi: 10.4236/jcc.2022.109001.