JURNAL EMACS
e-ISSN: 2686-2573
[Engineering, MAthematics and Computer Science] Vol.6 No.3 September 2024: 163-172 DOI: 10.21512/emacsjournal.v6i3.11904

American Sign Language Translation To Display the Text (Subtitles) Using a Convolutional Neural Network

Muhammad Fajar Ramadhan[1*], Samsuryadi[2], Anggina Primanita[3]

[1-3] Master of Computer Science, Faculty of Computer Science,
Sriwijaya University,
Palembang, Indonesia 30128
09012682226005@student.unsri.ac.id; samsuryadi@unsri.ac.id;
anggina.primanita@ilkom.unsri.ac.id

Correspondence: 09012682226005@student.unsri.ac.id

Abstract *- Sign language is a harmonious combination of hand gestures, postures, and facial expressions. One of the most used and also the most researched Sign Language is American Sign Language (ASL) because it is easier to implement and also more common to apply on a daily basis. More and more research related to American Sign Language aims to make it easier for the speech impaired to communicate with other normal people. Now, American Sign Language research is starting to refer to the vision of computers so that everyone in the world can easily understand American Sign Language through machine learning. Technology continues to develop sign language translation, especially American Sign Language using the Convolutional Neural Network. This study uses the Densenet201 and DenseNet201 PyTorch architectures to translate American Sign Language, then display the translation into written form on a monitor screen. There are 4 comparisons of data splits, namely 90:10, 80:20, 70:30, and 60:30. The results showed the best results on DenseNet201 PyTorch in the train-test dataset comparison of 70:30 with an accuracy of 0.99732, precision of 0.99737, recall (sensitivity) of 0.99732, specificity of 0.99990, F1-score of 0.99731, and error of 0.00268. The results of the translation of American Sign Language into written form were successfully carried out by performance evaluation using ROUGE-1 and ROUGE-L resulting in a precision of 0.14286, Recall (sensitivity) 0.14286, and F1-score. The more precise and sensitive the ROUGE results, the more accurate the real-time sign language translation will be.*

Keywords*: American Sign Language; Translation; Subtitles; DenseNet20; DenseNet201 PyTorch*

I. INTRODUCTION

Sign language became a colloquial language for people who were speechless. Previous researchers used sign language to communicate with people with disabilities. Sign language has always been a challenge until it finally becomes the language used in conversations with people in general (Delpreto et al., 2022; Malakan, 2021).

Problems of deafness or disability can occur from birth, or caused by an accident so that they cannot communicate properly. On this basis, many researchers are interested in overcoming or even treating deafness until now (Sensuse et al., 2022; Wei et al., 2023).

Previous research that has focused on the most about deafness is in the fields of health (Huang & Xia, 2020), social (Xu et al., 2023), and education (Al-Fraihat et al., 2020). Recommendations from this field show that the technology field is becoming very effective and relevant because technology is always experiencing development and massive around the world (Sundar & Bagyammal, 2022).

One of research (Marjusalinah et al., 2021) has succeeded in classifying American sign language finger spelling with very high accuracy so that computers can recognize American sign language. This research should be continued to the next stage, which is to display the writing on the monitor as previously classified. This study uses DenseNet121 as one of the convolutional Neural Network architectures used.

Among the sign languages that are growing in the world, American sign language is one of the many sign languages used around the world (Gurbuz, 2020). One of

the advantages of this sign language is that its use is from local to foreign, there are also not many varieties so it is easy to remember (Abdullahi & Chamnongthai, 2022) recognition of similar ASL words confused translation algorithms, which lead to misclassification. In this paper, based on fast fisher vector (FFV).

Meanwhile, another researcher (Lu & Chuang, 2022) researched American Sign Language as a method for taking pictures and generating letters with machine learning. Machine learning has produced a wide variety of translations into other languages more accurately.

Over time, American Sign Language research has evolved into the application of deep learning. However, at the beginning of its development (Zhu et al., 2020), the introduction of American Sign Language using deep learning faced challenges regarding the need for an optimal feature extraction and pre-processing process, which required efficient time and more accurate translation.

Meanwhile, similar research (Xu et al., 2023) applies deep learning methods to image recognition and proves that deep learning methods continue to evolve to become more effective along with deep machine learning. The more data used and the training carried out, the more accurate the expected results will be.

One of the deep learning that is still used and continues to grow today is the Convolutional Neural Network (CNN) (Huang et al., 2017). CNN has become one of the many solutions in dealing with the problem of digital imagery, especially in American Sign Language translation. CNNs have a framework that has been conceptualized in such a way that the weight division, scaling, and displacement are relatively the same.

The development of American Sign Language translation technology, by using CNN in translation, has facilitated research in improving accessibility for people with disabilities who use American Sign Language as their primary language. Subtitles are very useful so that everyone can use them to learn American Sign Language (Prajwal et al., 2022)which has a very different signing alphabet (e.g., two-handed instead of one-handed.

DenseNet201 and DenseNet201 PyTorch is one of the CNN architectures that is still popular since 2014 until now in digital image processing or RGB pixels (Beal, 2021; Huang et al., 2017). This research is expected to produce high accuracy evaluation metric results so that it can accurately translate between actual data and prediction data.

So, this study intends to implement a convolutional neural network to display letters that are arranged into words or sentences so that they are worthy of being called subtitles. The purpose of the study was to show the results of data processing on the tested model that DenseNet201 PyTorch has the potential to be developed in American Sign Language translation because it has the best results in certain data splits. This is shown in the comparison of accuracy between the two models. And this study shows the success of the model displaying writing. It is hoped that it can be developed in future research for better American Sign Language translation results.

II. METHODS

This research starts from the previous research that has been summarized in the introduction. Furthermore, it will be explained starting from data preparation to evaluating the performance of the Convolutional Neural Network and American Sign Language translation in the form of subtitles.

2.1 Dataset

The dataset used comes from kaggle.com/grassknoted/asl-alphabet. The website contains images of hands demonstrating American Sign Language by class. Each class contains 3,000 images, making the total dataset available as many as 87,000. The class in question consists of 29 classes, which consist of letters in the alphabet, namely from the letters A to Z, plus the functions del (delete), nothing, and space.

This study uses 300 data for each class, so that a total of 8,700 data is ready to be processed for training and testing. This study also applies the 29 classes mentioned earlier.

2.2 DenseNet Architecture

As mentioned in the introduction, the method used is to use a Convolutional Neural Network. A convolutional neural network is a convolutional network that operates both linearly and non-linear (Ali & Kim, 2020). There are many types and architectures of convolutional neural networks, but this study uses DenseNet.

This study specifically uses DenseNet for three reasons. First, DenseNet has been proven to show high accuracy in previous studies (Marjusalinah et al., 2021) with an accuracy evaluation result of 0.95. Second, since 2016 since its first appearance, DenseNet is still feasible to use and is one of the architectures that continues to be developed for image research (Liang et al., 2022). Third, DenseNet architecture testing has been successful by using various objects, such as horses (Huang et al., 2017), cars, and palm patterns in the context of hand sign language (Abdullahi & Chamnongthai, 2022)recognition of similar ASL words confused translation algorithms, which lead to misclassification. In this paper, based on fast fisher vector (FFV).

DenseNet (Qin et al., 2023) is a layered architecture with specific map features. Map features are arranged like interconnected chains, so data will pass through all layers without exception. The map feature bypasses the batch normalization, ReLU, and 3x3 convolution functions within DenseNet. The resulting output will be repeated until the most optimal result is obtained (Kouvakis et al., 2024).

DenseNet introduces a solid block that takes the output based on the previous convolutional layer so that it produces an output feature that is then passed on to the next convolutional layer. In Figure 1, the convolutional layer is represented by x0, x1, x2, x3, and x4 with the input being

an image obtained from the dataset. Meanwhile, H1, H2, H3, and H4 have Batch Normalization (BN), rectified linear units (ReLU), Pooling, and Convolution (Conv) processes. This happens gradually until it produces the prediction expected by the research.

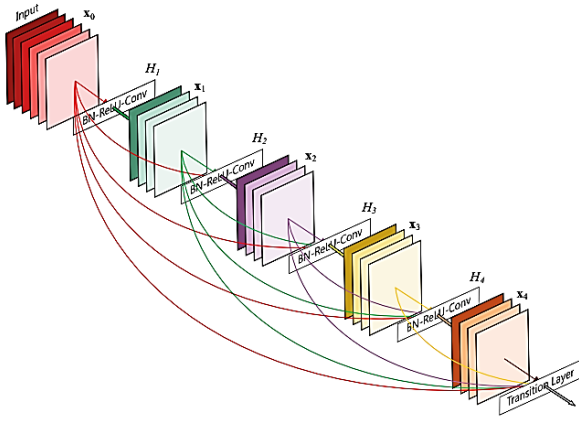


Figure 1. DenseNet Architecture

Based on Figure 1, there are five layer blocks where each layer has its own feature-maps process as input. Between the two layers, there is a convolution and pooling process with details like Table I (Huang et al., 2017)more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. In this paper, we embrace this observation and introduce the Dense Convolutional Network (DenseNet.

In this study, the DenseNet201 architecture use DenseNet201 and DenseNet201 PyTorch. It is called DenseNet201 because it has a number and size of 201 filters. Meanwhile, it is called PyTorch because it implements TorchVision on the DenseNet201 architecture. What distinguishes them is the library used (Huang et al., 2017).

In more detail, DenseNet201 has a relatively similar Output Size pattern. Starting from the convolution layer with an output size of 112 x 112. After the pooling process, the output size is compressed to 56 x 56 with 3 x 3 max pool and 2 strides. Stride can be said to be a filter shift. The process continues in the first Block Dense stage with a convolutional 1 x 1 and a transition layer with a 2 x 2 pooling average and stride 2. Pooling is also often called subsampling which is a reduction in the size of the matrix. Generally, what is often used is average pooling by taking the average convolutional value, while max pooling takes the maximum convolutional value (Bantupalli, 2019).

Table I. DenseNet201 Architecture

Layer name	Output size	Number and size of filters
Convolution layer	112x112	7x7 conv, stride 2
Pooling layer	56x56	3x3 max pool, stride 2
Block Dense (1)	56x56	1x1 conv
Transition layer (1)	56x56	[(1x1 conv) ¦ (3x3 conv)] x6
	28x28	2x2 avg pool, stride 2
Block Dense (2)	28x28	[(1x1 conv) ¦ (3x3 conv)] x12
Transition layer (2)	28x28	1x1 conv
	14x14	2x2 avg pool, stride 2
Block Dense (3)	14x14	[(1x1 conv) ¦ (3x3 conv)] x48
Transition layer (3)	14x14	1x1 conv
	7x7	2x2 avg pool, stride 2
Block Dense (4)	7x7	[(1x1 conv) ¦ (3x3 conv)] x32
Transition layer	1x1	7x7 global average pool
		1000D Fully Connected, Softmax

DenseNet201 Pytorch is an application of the DenseNet201 architecture based on PyTorch. Pytorch itself is a deep learning framework used to train and construct convolutional neural network models. DenseNet201 PyTorch has advantages such as reducing losses in gradient problems that DenseNet does not have in general, strengthening feature propagation by stimulating feature maps more optimally, and reducing the number of parameters substantially where more effective parameters for accuracy are maintained.

2.3 Preprocessing data

The dataset of 8,700 and 300 data for each class, the DenseNet201 and DenseNet201 PyTorch models will be train-tested with a ratio of 90:10, 80:20, 70:30, and 60:40, respectively. However, in the results and discussion, 1 best model from DenseNet201 and DenseNet201 PyTorch will be selected for the results of the confusion matrix (Saleh et al., 2024).

In this data preprocessing, the dataset is processed in such a way that the size is adjusted to 200x200 pixels of RGB image (Red, Green, Blue). The pixel range that should be 0-255 has been simplified to 0-1, making the initialization process faster and more efficient.

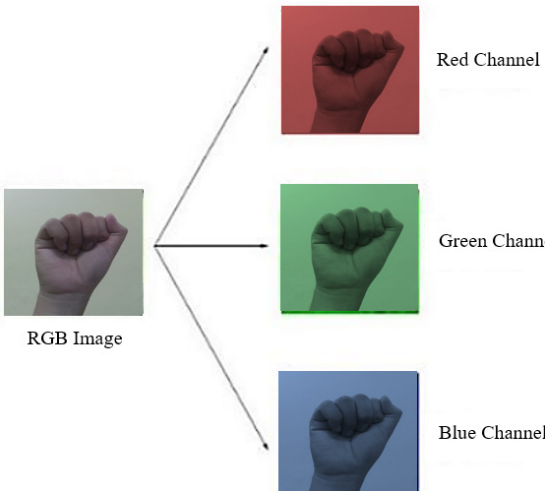


Figure 2. RGB Image

Figure 2: Image of RGB image in RGB format, screen image, and additive colors. RGB format for subtractive colors and colors not related to CYMK technology (cyan, magenta, yellow, and key (black) are subtractive colors that are available for unconventional colors, the most concise colors, the most intense colors, and the most important colors are attractive colors.

Classes that contain American Sign Language data in alphabets will be labeled so that they match between classes to labels. In this study, class 0 was labeled A, class 1 was labeled B, and so on. Then the last 3 classes in order are del (delete) for class 26, nothing for class 27, and space for class 28. Thus, the classes used are as many as 29 classes.

2.4 Performance Metric Calculation

This study calculates the quality of translation by paying attention to the evaluation of the accuracy performance of each model with a certain comparison. The parameters to be evaluated include accuracy, precision, specifications, recal / sensitivity, F1-score, and error as shown in Table II.

Table II. Formulas used for performance evaluation

Accuracy (A) = (TP+TN) / (TP+TN+FP+FN)	(2.1)
Precision (P) = TP / (TP+FP)	(2.2)
Recall (R) = TP / (TP+FN)	(2.3)
S = TN / (TN + FP)	(2.4)
$\mathbf{F_1}$ = 2 x (P x R) / (P + R)	(2.5)
E = 1 - A	(2.6)

The explanation of each performance evaluation in question. Accuracy (formula 2.1) calculates the percentage of correct predictions from the total predictions made by the model to provide an overview of the overall model performance. Precision (formula 2.2) is the ratio of correct predictions to total True Positive and False Positive predictions which is useful for reducing the number of diagnoses. Recall (formula 2.3) is the sensitivity or ratio of positive correct predictions to total positive data, which is important when the prediction is wrong as negative to minimize the number of incorrectly diagnosed negatives. The specification (formula 2.4) is the ratio of negative correct predictions to total actual negative data which helps identify the model's negative data precisely. F1-score (formula 2.5) is a linear average between precision and sensitivity, so it is very useful if there is a class imbalance in the dataset. Errors (formula 2.6) are the number of prediction errors made by the model to understand the areas where the model needs improvement. And finally, ROUGE (Recall-Oriented Understudy for Gisting Evaluation) on Table 2.4 is used to evaluate the quality of text summaries by comparing them with reference summaries that are very prevalent in natural language processing (NLP) or computer vision.

Then Table III shows the confusion matrix where TP is True Positive, FP is False Positive, TN is True Negative, and FP is False Negative. In addition to evaluating the performance of the classification model, the confusion matrix can evaluate the results of translation on the sign language of the research, especially in the evaluation of recall performance or sensitivity, precision, and F1-score which can be a reference for accuracy in the evaluation of the generated text. True Positive (TP) indicates the actual positive value that is correctly predicted as positive by the model. True Negative (TN) indicates the actual negative value that is correctly predicted as negative by the model. False Positive (FP) indicates the actual negative value that is incorrectly predicted as positive by the model. And False Negative (FN) shows the actual positive value that is predicted to be wrong as negative by the model.

Table III Table confusion matrix

		Prediction	
Truth		TP	FN
		FP	TN

The causes of prediction errors include images having high similarity so that the model fails to recognize patterns. Prediction errors are also caused by unique features in the class, unclear images, and poor resolution so that unique features become blurry and the model cannot predict the model correctly.

The evaluation of ROUGE's performance was carried out at a time after the translation of American Sign Language had been completed. Table IV is the formula applicable to the calculation of ROUGE. The ROUGE used is ROUGE-1 (for per-letter evaluation) and ROUGE L (Longest Common Subsequence) which is the longest row of references or candidates for writing.

ROUGE is an evaluation metric used to assess the quality of NLP (Natural Language Processing) tasks such as summarizing and translating text by machine learning. ROUGE measures the fit between a summary or translation generated by a model and a predetermined reference.

This study uses ROUGE-1 to measure the unigram-based conformity between the text generated by the model and the text stored in the dataset. Meanwhile, ROUGE-L measures the conformance based on the Longest Common Subsequence (LCS) between the text generated by the model and the text assigned to the dataset.

Table IV. Table used to ROUGE

Recall = (Number of matching n-grams) / (Number of n-grams in the Reference)	(2.7)
Precision = (Number of matching n-grams) / (Number of n-grams in the Candidate)	(2.8)
$\mathbf{R_{lcs}}$ = (LCS(X,Y)) / m	(2.9)
$\mathbf{P_{lcs}}$ = (LCS(X,Y)) / n	(2.10)
$\mathbf{F_{lcs}} = ((1+\beta^2)\, R_{lcs}\, P_{lcs}) / (R_{lcs}+\beta^2\, P_{lcs})$	(2.11)

As for the explanation for each formula, ROUGE-L is represented by the existing formula "LCS" which stands for Longest Common Subsequence.

Recall (formula 2.7) measures the number of unigrams of reference text that also appear in the model's resulting text. Precision (formula 2.8) measures the amount of text that the model results also appears in the reference text. F1-score is the average recall and precision that shows the balance between the two. Meanwhile, Recall LCS (formula 2.9) measures how long the longest subsequence of the reference text also appears in the same order in the model's resulting text. LCS precision (formula 2.10) measures how long the longest subsequence of the model's resulting text also appears in the same order in the reference text. And the F1-score LCS is a linear average between the recall LCS and the precision of the LCS indicating the balance between the two.

2.5 Proposed Method of American Sign Language Translation

Figure 3 shows that there are two parts to testing the model. The first part is a performance evaluation that results in an evaluation of the metrics of accuracy, precision, sensitivity, specificity, F1-score, error, and ROUGE. The second part is real-time data testing using a camera. The program used is a python program with the Pycharm platform. The program will translate American Sign Language according to the class, then display the results on the screen. The results of these two parts will be taken and conclusions will be drawn.

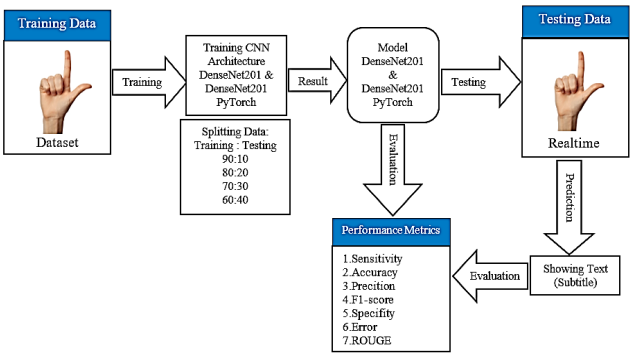


Figure 3. Proposed Method of American Sign Language Translation

Figure 3 is a proposed method that is summarized into six general stages that represent all the processes in this study.

- Train data on the available datasets. The processed dataset is 8,700 datasets with 29 classes consisting of alphabet letters and additional dels (Delete), nothing, and space.
- Dataset adjustment to the DenseNet201 and DenseNet201 PyTorch architectures used or called preprocessing. The dataset is adjusted to the running program at epoch 10, batch size 32, the dataset is adjusted to a size of 200 x 200 pixels with a range of 0 – 255 to 0.1. with variable 'A' fixed for class 1, 'B' for class 2, and so on.
- The results of preprocessing the running model are stored for the next stage, namely testing the model (number 4), and evaluating performance (number 6).
- The model test is carried out by translating hand sign language in real-time. To make predictions using openCV. program has been set up so that pressing the e keyboard to perform the ROUGE evaluation, the p keyboard to perform the prediction and the q keyboard to exit the American Sign Language translation program using openCV.
- Based on the number 4, any predicted American Sign Language translation by pressing the p keyboard will appear in the openCV in the upper left corner. The purpose of the study is to display subtitles with examples "H A L O M I K" for testing with ROUGE evaluates each letter and then an example of "HALO" with no pauses to show ROUGE evaluates live translation per word.
- After pressing the e keyboard for evaluation and the q keyboard to exit the openCV program, the program evaluates the recall performance, precision, and F1-score for the ROUGE-1 and ROUGE-L types. The results of this metric performance will be continued in the results and discussion in this study.

III. RESULT AND DISCUSSIONS

The DenseNet201 and DenseNet201 PyTorch architectural models have been created to translate American Sign Language by applying train-test split ratios of 90:10, 80:20, 70:30, and 60:40. This study displays the best results on each model, namely DenseNet201 with a ratio of 90:10, and DenseNet201 PyTorch with a ratio of 70:30 as shown in Table V.

Table V. Comparison of Accuracy Results between the two Architecture models

Dataset Split	Accuracy DenseNet201	Accuracy DenseNet201 PyTorch
90:10	0.98046	0.92874
80:20	0.95862	0.96552
70:30	0.95326	0.99732
60:40	0.95862	0.85374

Table V shows that the best DenseNet201 accuracy.

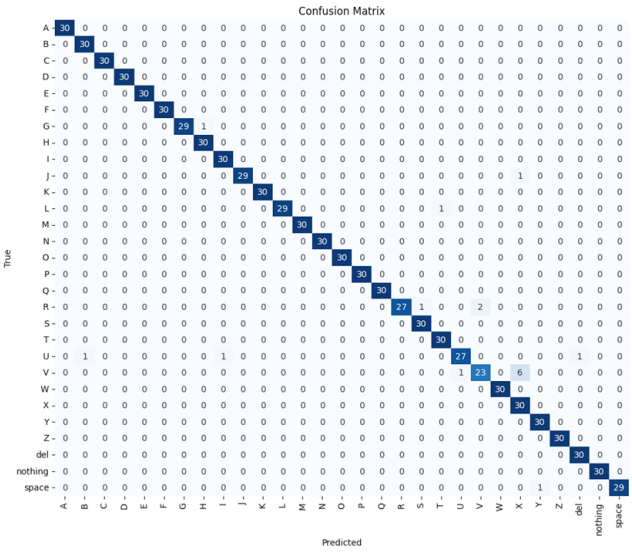


Figure 4 Confusion Matrix DenseNet201 architecture on 90:10 Dataset

To ensure that the accuracy of the model shows significant results, graphs are created during data preprocessing. This is done by tuning the parameters of batch 32 and epoch 10.

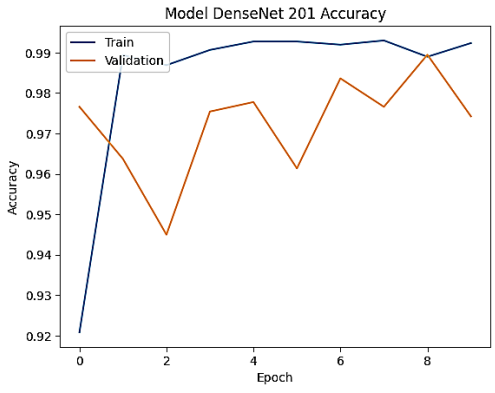


Figure 5. DenseNet201 Accuracy Metric Evaluation Graph 90:10 ratio

Figure 5 shows the accuracy evaluation metric of DenseNet201 with a ratio of 90:10. The training results showed high average results compared to the expected validation.

Figure 6. DenseNet201 Loss Metric Evaluation Graph 90:10 ratio

Figure 6 show the results of the graph where the training results are above the average validation value for accuracy, and show a slight loss. It can be analyzed that the DenseNet201 model should be able to translate American Sign Language accurately. Table VI is a table for the results of the evaluation of architectural performance with an approach to all classes that have been created.

Table VI. Results of DenseNet201 Architecture Performance Evaluation on DenseNet201 90:10

Class	Accuracy	Precision	Sensitivity	F1-Score	Specificity	Error
A	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
B	0.99885	0.96774	1.00000	0.98361	0.99881	0.00115
C	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
D	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
E	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
F	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
G	0.99885	1.00000	0.96667	0.98305	1.00000	0.00115
H	0.99885	0.96774	1.00000	0.98361	0.99881	0.00115
I	0.99885	0.96774	1.00000	0.98361	0.99881	0.00115
J	0.99885	1.00000	0.96667	0.98305	1.00000	0.00115
K	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
L	0.99885	1.00000	0.96667	0.98305	1.00000	0.00115
M	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
N	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
O	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
P	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
Q	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
R	0.99655	1.00000	0.90000	0.94737	1.00000	0.00345
S	0.99885	0.96774	1.00000	0.98361	0.99881	0.00115
T	0.99885	0.96774	1.00000	0.98361	0.99881	0.00115
U	0.99540	0.96429	0.90000	0.93103	0.99881	0.00460
V	0.98966	0.92000	0.76667	0.83636	0.99762	0.01034
W	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
X	0.99195	0.81081	1.00000	0.89552	0.99167	0.00805
Y	0.99885	0.96774	1.00000	0.98361	0.99881	0.00115
Z	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
del	0.99885	0.96774	1.00000	0.98361	0.99881	0.00115
nothing	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
space	0.99885	1.00000	0.96667	0.98305	1.00000	0.00115
Average	0.98046	0.98170	0.98046	0.98027	0.99930	0.01954

Based on Table VI, the average architectural performance evaluation was obtained with an accuracy of 0.98046, precision of 0.98170, sensitivity of 0.98046, specificity of 0.99930, F1-score of 0.98027, and error of 0.01954.

3.2 DenseNet201 PyTorch Performance Evaluation Results

Unlike the previous DenseNet201 which got the best performance evaluation results at a ratio of 90:10, in DenseNet201 PyTorch got the best performance evaluation results at a ratio of 70:30. Of the 8,700 datasets, this study uses 6,090 RGB image datasets as training data and another 2,610 as evaluation data to produce model performance. Tuning parameters are still the same as DenseNet201, namely batch size 32 and epoch 10.

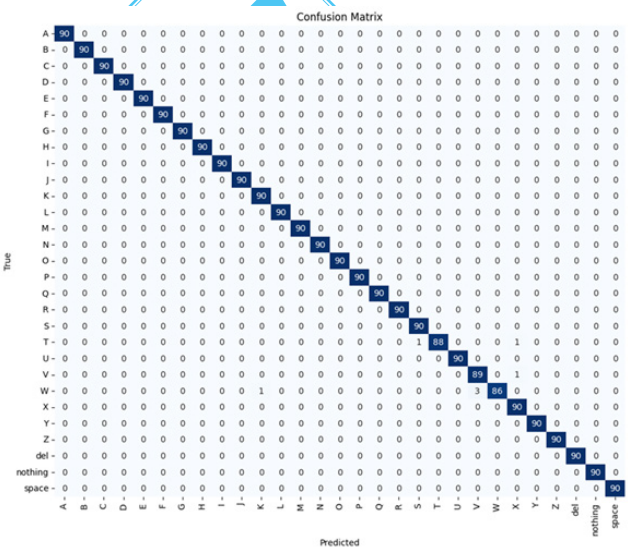


Figure 7. Confusion Matrix architecture DenseNet201 PyTorch at 70:30 Dataset

Figure 7 shows that there are only 3 classes where mistakes occurred, namely classes T, V, and W. class W showed the most erroneous results but no more errors than the previous DenseNet201 ratio of 90:10. Based on this, it can be analyzed that class W has a mistake so that it translates class was class K and V. and as is known, the letter W has a closeness or similarity to class V. Likewise, other prediction errors in the letter T that mispredict T with S and X and the letter V that mispredict V with the letter X.

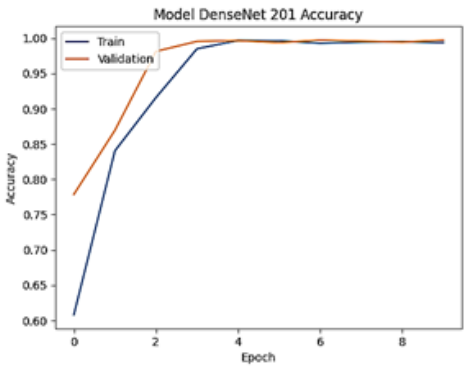


Figure 8. DenseNet201 PyTorch Accuracy Metric Evaluation Graph 70:30 ratio

Figure 8 shows a performance evaluation metric with high validation, but also shows balanced training results.

Based on Figure 8, DenseNet201 PyTorch shows high validation to show the accuracy results that a model should achieve. In contrast to the previous DenseNet201 which tended to go up and down or was called unstable. However, DenseNet201's 70:30 ratio showed stable training results in the fourth epoch.

Table VII. Results of DenseNet201 Architecture Performance Evaluation on DenseNet201 PyTorch rasio 70:30

Class	Accuracy	Precision	Sensitivity	F1-Score	Specificity	Error
A	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
B	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
C	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
D	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
E	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
F	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
G	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
H	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
I	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
J	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
K	0.99962	0.98901	1.00000	0.99448	0.99960	0.00038
L	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
M	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
N	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
O	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
P	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
Q	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
R	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
S	0.99962	0.98901	1.00000	0.99448	0.99960	0.00038
T	0.99923	1.00000	0.97778	0.98876	1.00000	0.00077
U	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
V	0.99847	0.96739	0.98889	0.97802	0.99881	0.00153
W	0.99847	1.00000	0.95556	0.97727	1.00000	0.00153
X	0.99923	0.97826	1.00000	0.98901	0.99921	0.00077
Y	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
Z	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
del	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
nothing	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
space	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000
Average	0.99732	0.99737	0.99732	0.99731	0.99990	0.00268

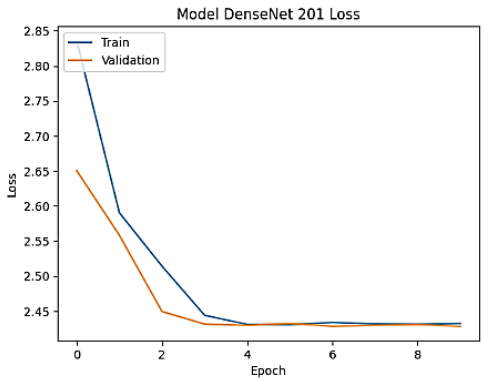


Figure 9. DenseNet201 PyTorch Loss Metric Evaluation Graph 70:30 ratio

Based on Figure 9, Densenet201 Pytorch has almost no loss, as Figure 9 tends to have very high accuracy. This shows that between very high accuracy results and very low loss values show balanced results.

Table VII, the performance results for the DenseNet201 PyTorch architecture model with accuracy of 0.99732, precision of 0.99737, sensitivity of 0.99732, specificity of 0.99990, F1-score of 0.99731, and error of 0.00268 were obtained. This shows that the DenseNet201 Pytorch with a 70:30 ratio is superior to the DenseNet201 architecture with a 90:10 ratio, especially in the ratio of the number of classes that can be predicted precisely.

3.3 Results of the Performance Evaluation of American Sign Language Translation

Next is to test the model that has been made with American Sign Language translation in real time. Based on Table VIII, the model was successfully predicted and displayed on the monitor. The shooting on Table 2.8 was carried out in a closed room with a blue background, using a device camera whose results were taken in real-time. The hand gestures demonstrated are in accordance with the rules that apply to the use of American Sign Language. Live camera translates American Sign Language in the form of subtitles.

Table VIII. Results of American Sign Language Translation in "HALO MIK" Hand Gestures

Hand Gestures	Word & Prediction Results
	H = Q
	A = E
	L = Y
	O = O

Hand Gestures	Word & Prediction Results
	*Space* = *Space*
	M = S
	I = T
	K = V

The limitation and challenge in this section is that the model successfully translates American Sign Language, even though it has not been predicted correctly. This is evidenced by the results of the evaluation of the performance of American Sign Language translation that has been successfully carried out and displays subtitles on the monitor screen. Based on Table VIII, the correctly translated American Sign Language is class O and class space, while there are other errors in translating American Sign Language.

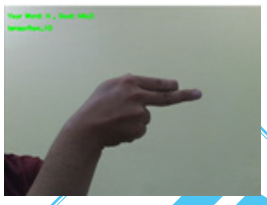
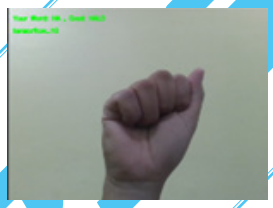
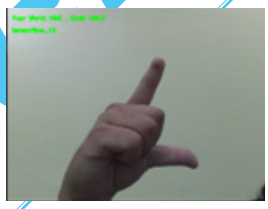
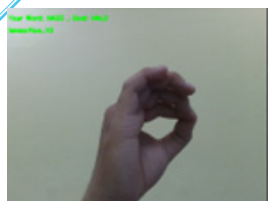
Table IX. Results of ROUGE's Performance Evaluation on American Sign Language Translation

Evaluation	*Precision*	*Recall (Sensitivity)*	F1-*score*
ROUGE-1	0.14286	0.14286	0.14286
ROUGE-L	0.14286	0.14286	0.14286

Table IX, the performance evaluation results of ROUGE-1 are the same as ROUGE-L, which both show results with a precision of 0.14286, Recall 0.14286, and F1-score 0.14286. A precision of 0.14286 indicates that the translation has measured the number of model-generated texts that appear in the prediction text by one letter (0.14286). Recall 0.14286 indicates that the translation has measured the amount of reference text that also appears in the model's resulting text, which in this case is equal to 0.14286. and F1-score shows the average recall and precision results where both produce the same evaluation value.

The study has been conducted as the results of the translation of American Sign Language in Table VIII and the evaluation of ROUGE performance in Table IX. The program is made with a space break for each letter in the subtitle so that the accuracy produced is in accordance with the expected letter (goal).

Table X. Results of American Sign Language Translation in "HALO" Hand Gestures

Hand Gestures	Word & Prediction Results
	H = H
	A = A
	L = G
	O = O

In another example, Table X shows fairly accurate results, although there is still a mistake in the letter L that translates the letter L as G. The study was also carried out, but did not take the ROUGE evaluation because the result was 0 (zero), because the subtitles did not meet the goal. This is because the expected prediction results are different from the expected goals in the model. In addition, with a green background and sufficient lighting, it turns out to affect the results significantly, although there are still mistakes in the program.

3.4 Discussion

Based on the results of the entire research process, the DenseNet201 PyTorch architecture model with a 70:30 ratio obtained performance evaluation results with an accuracy of 0.99732 (99.73%) while DenseNet201 had a 90:10 ratio of 0.98046 (98.05%). The advantages of the DenseNet201 Pytorch Model with a 70:30 ratio can be seen in the results of the confusion matrix which has a slight class prediction error compared to the DenseNet201 Model with a ratio of 90:10. In detail, DenseNet201 PyTorch 70:30 ratio obtained performance evaluation results with accuracy of 0.99732, precision 0.99737, recall (sensitivity) 0.99732, specificity 0.99990, F1-score 0.99731, and error 0.00268.

Then, the results of American Sign Language translation have been successfully applied with the architectural model that has been made. However, there is still confusion in the translation of American Sign Language. There are several factors that cause confusion in American Sign Language translation. First, it is caused by a lack of lighting when shooting using a camera. Second, the limited camera resolution causes blur in the results of the American Sign Language translation. Third, because the dataset used is limited, there is a lack of model references for translating American Sign Language.

In addition, the significant difference between datasets and American Sign Language in real-time was very influential in this study. Both DenseNet201 with a 90:10 ratio and DenseNet201 PyTorch with a 70:30 ratio get the same ROUGE results, both ROUGE-1 and ROUGE-L, namely precision 0.14286, recall (sensitivity) 0.14286, and F1-score 0.14286.

Overall, this study still shows limitations that can be used as the main reference to be improved in future research. First, this study used 8,700 out of 87,000 datasets from kaggle, and as American Sign Language develops, an increase in the number of datasets is very likely. The reduction of the dataset aims to speed up research where using 87,000 with limited devices can take twenty-four hours for one epoch or more. Second, the epoch applied is ten out of a hundred due to the limited research time. Epoch is the process of training on a convolutional neural network for one round. It is hoped that future research can apply a larger number of epochs to get optimal results. Third, the difference in the dataset in Kaggle to American Sign Language in real time. Further research is expected to create a dataset with more references, such as using red, green, or blue backgrounds, dark, dim, well-lit, and dazzling, and the clarity or blurry of the images to be used as datasets also affect the results of future research.

IV. CONCLUSION

The research on American Sign Language translation to display subtitles has obtained the best model of DenseNet201 PyTorch architecture with a ratio of 70:30 with an accuracy of 0.99732, a precision of 0.99737, a recall (sensitivity) of 0.99732, a specificity of 0.99990, an F1-score of 0.99731, and an error of 0.00268. Meanwhile, the results of the American translation were successfully carried out and the performance evaluation of ROUGE-1 and ROUGE-L with a precision of 0.14286, Recall (sensitivity) 0.14286, and F1-score 0.14286.

The results of the evaluation of the performance of American Sign Language that show high accuracy results have an impact on future research to use the same architectural model, or to use another architectural model with the same or better performance evaluation results. The better the results of American Sign Language translation, the more impactful it will have on the implementation of American Sign Language translation applied around the world. This has also had an impact on the technological advances that have developed against other similar studies, such as the translation of facial cues or even body sign language.

Future work for this research is expected to use more and better datasets, because datasets are very important so that the model can translate American Sign Language better. Furthermore, the appearance of subtitles is expected to be continued to the next level, such as being able to capture American Sign Language in various environments that have complex situations and conditions. It is hoped that American Sign Language translation will get more accurate real-time results so that the communication process between users becomes easier and without certain limitations.

REFERENCES

Abdullahi, S. B., & Chamnongthai, K. (2022). American Sign Language Words Recognition Using Spatiooral Prosodic and Angle Features: A Sequential Learning Approach. *IEEE Access*, *10*, 15911–15923. https://doi.org/10.1109/ACCESS.2022.3148132

Bantupalli, K. (2019). American Sign Language Recognition using Deep Learning and Computer Vision. In *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018* (pp. 4896–4899). https://doi.org/10.1109/BigData.2018.8622141

Delpreto, J., Hughes, J., D'Aria, M., De Fazio, M., & Rus, D. (2022). A Wearable Smart Glove and Its Application of Pose and Gesture Detection to Sign Language Classification. *IEEE Robotics and Automation Letters*, *7*(4), 10589–10596. https://doi.org/10.1109/LRA.2022.3191232

Gurbuz, S. Z. (2020). A linguistic perspective on radar micro-doppler analysis of American sign language. In *2020 IEEE International Radar Conference, RADAR 2020* (pp. 232–237). https://doi.org/10.1109/RADAR42522.2020.9114818

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 2261–2269. https://doi.org/10.1109/CVPR.2017.243

Kouvakis, V., Trevlakis, S. E., & Boulogeorgos, A. A. A. (2024). Semantic Communications for Image-Based Sign Language Transmission. *IEEE Open Journal of the Communications Society*, *5*(January), 1088–1100. https://doi.org/10.1109/OJCOMS.2024.3360191

Malakan, Z. M. (2021). Classify, Detect and Tell: Real-Time American Sign Language. In *Proceedings - 2021 IEEE 4th National Computing Colleges Conference, NCCC 2021*. https://doi.org/10.1109/

NCCC49330.2021.9428808

Marjusalinah, A. D., Samsuryadi, S., & Buchari, M. A. (2021). Classification of Finger Spelling American Sign Language Using Convolutional Neural Network. *Computer Engineering and Applications Journal*, *10*(2), 93–103. https://doi.org/10.18495/comengapp.v10i2.377

Prajwal, K. R., Bull, H., Momeni, L., Albanie, S., Varol, G., & Zisserman, A. (2022). Weakly-supervised Fingerspelling Recognition in British Sign Language Videos. *BMVC 2022 - 33rd British Machine Vision Conference Proceedings*, 1–19.

Saleh, A. B. U., Miah, M., & Hasan, A. L. M. (2024). Sign Language Recognition Using Graph and General Deep Neural Network Based on Large Scale Dataset. *IEEE Access*, *12*(January), 34553–34569. https://doi.org/10.1109/ACCESS.2024.3372425

Sensuse, D. I., Putro, P. A. W., Rachmawati, R., & Sunindyo, W. D. (2022). Initial Cybersecurity Framework in the New Capital City of Indonesia: Factors, Objectives, and Technology. *Information (Switzerland)*, *13*(12), 1–10. https://doi.org/10.3390/info13120580

Wei, W., Wang, J., Li, J., & Xu, M. (2023). A novel image recommendation model based on user preferences and social relationships. *Journal of King Saud University - Computer and Information Sciences*, *35*(7), 101640. https://doi.org/10.1016/j.jksuci.2023.101640