International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 6 Issue 8, August - 2022, Pages: 25-34

Classification of Sign-Language Using Deep Learning by ResNet

Tanseem N. Abu-Jamie, Pror.Dr. Samy S. Abu-Naser

Department of Information Technology,
Faculty of Engineering and Information Technology,
Al-Azhar University - Gaza, Palestine

Abstract*: American Sign Language, or ASL as its acronym is commonly known, is a fascinating language, and many people outside of the Deaf community have begun to recognize its value and purpose. It is a visual language consisting of coordinated hand gestures, body movements, and facial expressions. Sign language is not a universal language; it varies by country and is heavily influenced by the native language and culture. The American Sign Language alphabet and the British Sign Language alphabet are completely contrary. Fingerspelling is one-handed in ASL and two-handed in BSL, with the exception of the letter C. AI technologies, particularly deep learning, can play an important role in breaking down communication barriers between deaf or hearing-impaired people and other communities, significantly contributing to their social inclusion. Recent advances in sensing technologies and AI algorithms have paved the way for the development of a wide range of applications aimed at meeting the needs of the deaf and hearing-impaired communities. To that end, the purpose of this paper is By attempting to translate sign language using artificial intelligence algorithms, we focused in this paper on a transfer learning technique based on deep learning using a ResNet algorithm and compared it to previous papers in which we used VGG19[1]and mobileNet [2], where we get in the ResNet algorithm on the degree of Accuracy 93,48% e number of images (43500 in the dataset in size 64*64 pixel) and the data split training data into training dataset (70%) and validation dataset(15%) and testing dataset(15%) and 20 epoch.*

Keywords: *sign language Classification, Type of sign language, Deep Learning, Classification, Detection*

1. INTRODUCTION:

Communication is a skill required for community members to express themselves and live in harmony. While the members of the community can communicate using a verbal and auditory language, those who do not have these skills can express themselves using sign language, which is a visual language. Sign language is a communication tool in which people with hearing disabilities or difficulties communicate using body movements such as hands, arms, and gestures. Because most people in our society are unfamiliar with the sign language used by people with hearing disabilities to communicate and express themselves, it is evident that these people have difficulty expressing themselves in their daily lives.

The research community has long recognized this need developing sign language technologies to help hearing-impaired people communicate and socialize. Although the development of such technologies can be difficult due to the existence of numerous sign languages and a lack of large annotated datasets, recent advances in AI and machine learning have played an important role in automating and improving such technologies.[3]

The development of powerful machine learning algorithms to robustly classify human articulations to isolated signs or continuous sentences is the goal of sign language recognition (SLR). The lack of large annotated datasets currently limits SLR methods' accuracy and generalization ability, as well as the difficulty in identifying sign boundaries in continuous SLR scenarios.

2. REVIEW OF LITERATURE SURVEY

Several researchers have worked in the field of sign language recognition over the last two decades. The input captured and fed into the system is used to classify static sign language recognition systems. The literature review in this work focuses not only on this category, but also on the use of different classifiers and their application in Machine Learning and Deep Learning-based systems. [4]

In this paper [5] they present a malware family classification approach using a deep neural network based on the ResNet-50 architecture. Malware samples are represented as byteplot grayscale images, and a deep neural network is trained by freezing the pre-trained convolutional layers of ResNet-50 on the ImageNet dataset and adapting the final layer to malware family classification. The experimental results on a dataset of 9,339 samples from 25 different families demonstrated that our approach can effectively classify malware families with a 98.62 % accuracy.

In this paper [6] They propose a method for achieving scene level classification of high spatial resolution images in the absence of a large number of training samples. They used a residual learning network (ResNet) to extract depth features from high resolution remote sensing images, as well as low-level features such as color moment features and gray level co-occurrence matrix features. They used these to construct various scene semantic features of high-resolution images, and then used the training

support vector machine to create a classification model (SVM). Using the UC Merced Land Use (UCM) data set as the migration sample, the scene classification accuracy of the GF-2 data set can reach 95.71 % with a small sample size, according to the sample migration method. Finally, GF-2 image scene level classification is implemented in accordance with reality using this method.

In this paper [7] They propose a new model called Global Average Pooling Residual Network (G-ResNet) to classify brain tumor images. The model has the following features: (1) For the classification task, we use the well-known CNN architecture in deep learning known as ResNet34. (2) For classification, we use the global average pooling layer rather than the flattened layer to reduce the number of parameters and avoid overfitting. (3) To improve classification accuracy, we concatenate the feature vectors of different layers in order to fuse the network's low-level and high-level features. (4) We define a loss function as the sum of the interval and cross entropy losses. The penalty for misclassification is increased by the total loss. In summary, the model achieves 95.00 % classification accuracy, which is significantly higher than the previous models.

In [8] In this study, classification processes on images were carried out by using CapsNet, AlexNet and ResNet-50, DenseNet, VGG16, Xception, InceptionV3, NasNet, EfficentNet, Hitnet, Squeezenet architectures and TSLNet, which was designed for the study. When the deep learning models were examined, it was found that CapsNet and TSLNet models were the most successful models with 99.7% and 99.6% accuracy rates, respectively.

In [9] their deep network is proposed to be used for their sign language proposed network was able to achieve an accuracy of 90.3 % which is comparable to other works including those that used depth images in addition to RGB images. their network was also able to achieve prediction rates of 50 to 100 Hz which makes it capable of real-time prediction.

In [10] this paper, Gesture recognition is proposed of static ASL using Deep Learning. The contribution consists of two solutions to the problem. The first one is resized with Bicubic static ASL binary images. Besides that, good recognition results in of detection the boundary hand using the Robert edge detection method. The second solution is to classify the 24 alphabets static characters of ASL using Convolution Neural Network (CNN) and Deep Learning. The classification accuracy equals to 99.3 % and the error of loss function is 0.0002. According to 36 minutes with 15 seconds of elapsed time result and 100 iterations. The training is fast and gives the very good results, in comparison with other related works of CNN, SVM, and ANN for training.

Based on improved MobileNetV1, this paper [11] proposed an image recognition method for defective button batteries. MobileNetV1's deep convolution layer was improved, and the Relu activation function was replaced by the more stable TanhExp activation function. The experimental results show that the proposed method performs exceptionally well in detecting defective button battery images. The test set's accuracy can reach 99.05 percent, allowing it to be used to identify imperfect button battery images.

The paper [12] proposes three hybrid MobileNet architectures which has improved accuracy along-with reduced size, lesser number of layers, lower average computation time and very less overfitting as compared to the baseline MobileNet v1. The reason behind developing these models is to have a variant of the existing MobileNet model which will be easily deployable in memory constrained MCUs.the name the model having the smallest size (9.9 MB) as Thin MobileNet. there achieve an increase in accuracy by replacing the standard non-linear activation function ReLU with Drop Activation and introducing Random erasing regularization technique in place of drop out. The model size is reduced by using Separable Convolutions instead of the Depthwise separable convolutions used in the baseline MobileNet.

3. METHODOLOGY

The proposed system for Sign Language Recognition System comprises of data to Kaggle the dataset contains sets of images in jpeg extension and prepare data to apply the architecture CNN by resNet model data split training data into training dataset (70%) and validation dataset (15%) and testing dataset(15%) Training and testing the dataset

A. Dataset

The proposed system employs deep learning techniques especially the network ResNet for data recognition in the training phase, which is critical to the overall system. We used just 87,000 images because the dataset is very large . This data contains 29 classes (A to Z), spaces, delete, and nothing .

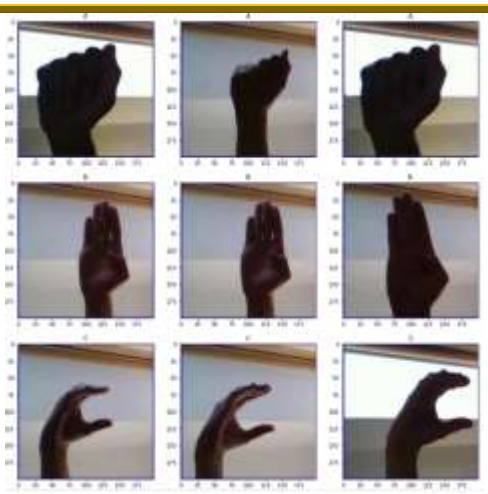

Figure 1:some of the latters used in dataset

B. Deep learning vs. machine learning in artificial intelligence

To understand deep learning, machine learning and artificial intelligence, the following definitions is important to know:

1. A method called artificial intelligence (AI) enables computers to simulate human intelligence. There is machine learning in it.
2. A subset of artificial intelligence called machine learning uses methods (like deep learning) that allow machines to learn from experience and become better at completing tasks. The following steps form the foundation of the learning process:
    1. Put information into an algorithm. (During this step, you can provide the model more data by performing feature extraction, for instance.)
    2. To train a model, use the data below.
    3. To deploy and test the model.
    4. Utilize the deployed model to carry out a prediction task automatically. (Or, call the deployed model and utilize it to get the predictions it has returned.)
3. A branch of machine learning called "deep learning" uses artificial neural networks as its foundation. Because artificial neural networks have multiple input, output, and hidden layers, the learning process is complex. Each layer has components that convert the incoming data into knowledge that the following layer can utilize to perform a specific predicted task. This structure enables a machine to learn by processing its own data.[13]

we can create computer programs and systems that do activities that are frequently attributed to human intellect by utilizing machine learning and deep learning approaches. These include language translation, speech recognition, and picture and image recognition. The suggested system's objective is to identify the sign that is denoted by the sign, and 29 signs were used in the experiment.

C. ResNet

Deep residual networks are convolutional neural networks (CNNs) with more than 50 layers, like the well-known ResNet-50 model. A Residual Neural Network (ResNet) is a type of Artificial Neural Network (ANN) that builds a network by piling residual blocks on top of one another.

Residual Network is referred to as ResNet. In their 2015 computer vision research paper titled "Deep Residual Learning for Image Recognition,"initially introduced this artificial neural network.[14]

There are other versions of ResNet that use the same basic idea but have various numbers of layers. The form that can operate with 50 neural network layers is referred known as Resnet50.

Machine learning specialists add extra layers while using deep convolutional neural networks to address a computer vision challenge. Because the various layers may be trained for a variety of jobs to produce highly precise results, these additional layers aid in the more effective solution of complicated issues.

Although the number of stacked layers might enhance the model's features, a deeper network can reveal the degradation

problem. In other words, as the neural network's layer count rises, accuracy levels may eventually get saturated and begin to slowly deteriorate. As a result, the model's performance declines on both the training data and the testing data. Overfitting did not cause this degradation.

ResNet was built with the goal of solving this specific problem. Remaining blocks are used in deep residual networks to boost model precision. The basis of this kind of neural network is the idea of "skip connections," which is at the foundation of the residual blocks.

These skip connections operate in two different ways. First, they resolve the problem of the vanishing gradient by creating a different shortcut for the gradient to use. They also give the model the ability to learn an identity function. This ensures it is tried to ensure that the model's higher levels don't perform any worse than its lower layers. In summary, the layers learn identity functions much more quickly thanks to the residual blocks. ResNet hence reduces the number of errors while increasing the effectiveness of deep neural networks with more neural layers. To put it another way, the skip connections combine the results of earlier layers with the results of stacked layers, enabling the training of far deeper networks than was previously feasible.

The following component is part of the Resnet 50 architecture:

1. A convolution with 64 different kernels, each with a stride of size 2, and a kernel size of 7 * 7 gives us 1 layer.
2. Then, we see maximum pooling with a 2 stride size.
3. The following convolution consists of three layers: a 1*1,64 kernel, a 3*3,64 kernel, and finally a 1*1,256 kernel. These three levels are repeated a total of three times, giving us nine layers in this phase.
4. The kernel of 1 * 1,128 is shown next, followed by the kernel of 3 * 3,128 and, finally, the kernel of 1 * 1,512. We performed this procedure four times for a total of 12 layers.
5. Following that, we have a kernel of size 1 * 1,256, followed by two more kernels of size 3 * 3,256 and size 1 * 1,1024; this is repeated six times, giving us a total of 18 layers.
6. Finally, a 1 * 1,512 kernel was added, followed by two more kernels of 3 * 3,512 and 1 * 1,2048. This process was done three times, giving us a total of nine layers.
7. Then we do an average pool, finish it with a completely linked layer made up of 1000 nodes, and add a softmax function to produce one layer.
8. The activation operations and the maximum/average pooling layers are not actually counted.

So when we add everything up, we get a 50-layer Deep Convolutional network with 1 + 9 + 12 + 18 + 9 + 1 layers.

How to utilize Keras and ResNet 50

Step #1: In order to convert the CNN into a residual network and construct the convolution block, you must first run a code to specify the identity blocks.

Step #2: The next step is to combine both blocks to create the 50-layer Resnet model.

Step #3: The final step is to teach the model to perform the necessary task. You may quickly create a thorough overview of the network design you created with Keras. For later usage, this can be printed or saved

D. Training Algorithm

The CNN ResNet model is utilized in Table 1and preparing pictures size with a 64x64x3 pixel resolution. The learning rate was set to 0.0001, the activation to softmax, the loss to categorical crossentropy, the optimizer to Adam, and the epoch to 20. These parameters were used to fit the model.

Table 1:trainig algorithem

Model	Number of image	testing Time (seconds)	Accuracy (Percent)
ResNet	29000	5.4411s	99,87%

4. EXPERIMENTAL RESULTS

1. load all images in memory

The library NumPy was invoked first. We utilized Numpy.random.seed in this module () The pseudo-random number generator in Python receives its input from the np.random.seed method. [11] We apply it to Random sampling is necessary for simple activities like dividing datasets into training and test sets. Pseudorandom numbers are also almost always needed for random sampling.

Additionally, we select the top 1000 photographs for each class using a filter before resizing them to (64*64 px). The outcome

of this phase is thus:

1. size of the training data: (29000, 64, 64, 3)
2. Training labels shape: (29000, 29)

2. **Split dataset:** Import train test split from sklearn.model selection to split training data into a training dataset (70%), validation dataset (15%), and testing dataset (15%).
3. **call the ResNet pre-trained Model**
   we make function contain a ResNet model use(weights= 'imagenet', include_top=False, input_shape= (64,64,3)) then make MaxPooling2D and Dense then used compile (Adam(lr=0.00001), loss='categorical_crossentropy', metrics=['accuracy'])
   So the result in this step :

1. Total params: 23,647,133
2. Trainable params: 23,594,013
3. Non-trainable params: 53,120

4. **Train model**

Fit_generator is frequently used when datasets are too huge to fit into memory, and we used it during training. They are frequently difficult as well, necessitating data augmentation to prevent overfitting and improve the generalizability of our model.

The results of a convolution neural network's by ResNet loss rate in the training and test sets after 20 repetitions are shown in Figure 8. Which suggests that the convolutional neural network successfully learned the input and can act as a good model for understanding sign language but VGG16 100% in the previous paper [1] mobileNet 95,41% accuracy[2] but was the ResNet 99.78%

```
Epoch 1/20
61/61 [==============================] - 38s 263ms/step - loss: 1.3398 - accuracy: 0.6902 - val_loss: 1.8930 - val_accuracy: 0.5683
Epoch 2/20
61/61 [==============================] - 14s 223ms/step - loss: 0.1424 - accuracy: 0.9627 - val_loss: 0.4342 - val_accuracy: 0.8754
Epoch 3/20
61/61 [==============================] - 16s 266ms/step - loss: 0.0513 - accuracy: 0.9848 - val_loss: 0.1347 - val_accuracy: 0.9595
Epoch 4/20
61/61 [==============================] - 16s 256ms/step - loss: 0.0278 - accuracy: 0.9915 - val_loss: 0.0495 - val_accuracy: 0.9854
Epoch 5/20
61/61 [==============================] - 12s 201ms/step - loss: 0.0252 - accuracy: 0.9932 - val_loss: 0.0743 - val_accuracy: 0.9819
Epoch 6/20
61/61 [==============================] - 12s 199ms/step - loss: 0.0293 - accuracy: 0.9919 - val_loss: 0.0536 - val_accuracy: 0.9868
Epoch 7/20
61/61 [==============================] - 14s 230ms/step - loss: 0.0196 - accuracy: 0.9937 - val_loss: 0.0203 - val_accuracy: 0.9928
Epoch 8/20
61/61 [==============================] - 12s 201ms/step - loss: 0.0177 - accuracy: 0.9945 - val_loss: 0.0283 - val_accuracy: 0.9928
Epoch 9/20
61/61 [==============================] - 12s 203ms/step - loss: 0.0187 - accuracy: 0.9950 - val_loss: 0.0231 - val_accuracy: 0.9925
Epoch 10/20
61/61 [==============================] - 14s 234ms/step - loss: 0.0139 - accuracy: 0.9956 - val_loss: 0.0115 - val_accuracy: 0.9975
Epoch 11/20
61/61 [==============================] - 12s 201ms/step - loss: 0.0150 - accuracy: 0.9967 - val_loss: 0.0172 - val_accuracy: 0.9957
Epoch 12/20
61/61 [==============================] - 16s 259ms/step - loss: 0.0075 - accuracy: 0.9988 - val_loss: 0.0047 - val_accuracy: 0.9986
Epoch 13/20
61/61 [==============================] - 12s 202ms/step - loss: 0.0076 - accuracy: 0.9985 - val_loss: 0.0335 - val_accuracy: 0.9910
Epoch 14/20
61/61 [==============================] - 12s 199ms/step - loss: 0.0160 - accuracy: 0.9965 - val_loss: 0.0186 - val_accuracy: 0.9930
Epoch 15/20
61/61 [==============================] - 12s 202ms/step - loss: 0.0062 - accuracy: 0.9981 - val_loss: 0.0053 - val_accuracy: 0.9980
Epoch 16/20
61/61 [==============================] - 16s 258ms/step - loss: 0.0069 - accuracy: 0.9988 - val_loss: 0.0039 - val_accuracy: 0.9986
Epoch 17/20
61/61 [==============================] - 14s 228ms/step - loss: 0.0044 - accuracy: 0.9990 - val_loss: 0.0045 - val_accuracy: 0.9979
Epoch 18/20
61/61 [==============================] - 14s 233ms/step - loss: 0.0037 - accuracy: 0.9994 - val_loss: 0.0021 - val_accuracy: 0.9992
Epoch 19/20
61/61 [==============================] - 12s 200ms/step - loss: 0.0058 - accuracy: 0.9988 - val_loss: 0.0099 - val_accuracy: 0.9969
Epoch 20/20
61/61 [==============================] - 12s 201ms/step - loss: 0.0114 - accuracy: 0.9978 - val_loss: 0.0143 - val_accuracy: 0.9962
Time elapsed in seconds: 354.75195240974426
```

Figure 2: Loss and Accuracy rate

5. Training and validation accuracy

the distinction between Validation dataset is the dataset used during the training process to validate the model's capacity to generalize or to stop it early. When we have created the model but want to use multiple datasets to validate it. The amount of precise data we have for training is typically a constraint. However, verifying a model is also required so that we may trust it

after evaluating it using a validation dataset.

Before performing tests on the training dataset, we evaluate the trained model on the validation dataset.

There are two methods to accomplish that:

1. Taking validation dataset from training dataset.
2. While dividing the primary dataset, keep a different validation set.

Training dataset is the data used to fit the model. Additionally, testing dataset refers to information utilized for purposes other than training and validating.

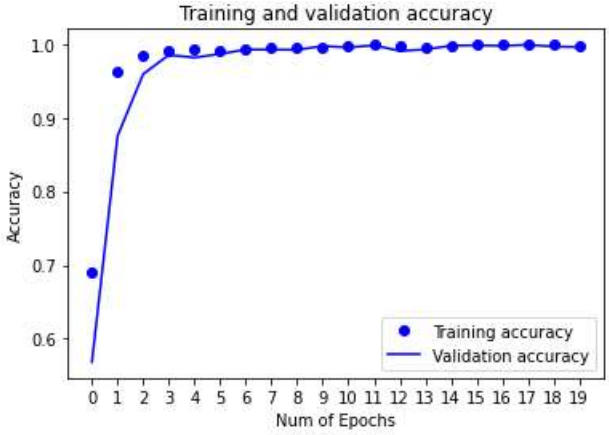

Figure 3: Training and validation accuracy

In Figure 7 show us no overfitting or underfitting this means that the model is well trained.

6. Training and validation loss

How "excellent" or "poor" a predictor is at categorizing the input data points in a dataset is quantified by a loss function.

The classifier does a better job of simulating the relationship between the input data and the output targets the smaller the loss.

However, there is a point where we can overfit our model by modeling the training data too closely, our model loses the ability to generalize.

The training loss is a metric that measures how well a deep learning model fits the training data. That is, it evaluates the model's error on the training set. and Validation loss, on the other hand, is a metric used to evaluate the performance of a deep learning model on the validation set. The validation set is a subset of the dataset set aside to test the model's performance.

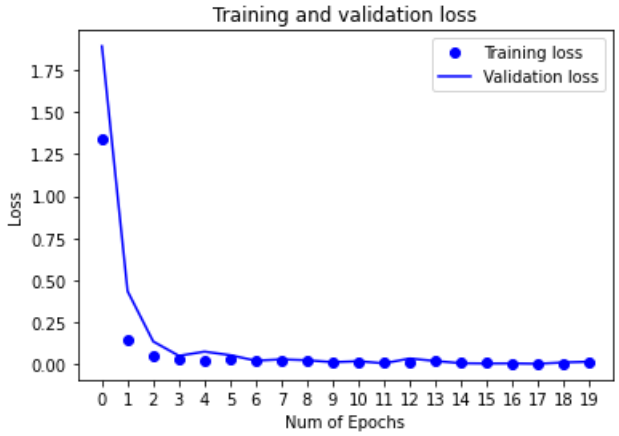

Figure 4: Training and validation loss

7. Evaluating the model on the training, validating sets

Training evaluation identifies training gaps and even opportunities for model improvement By seeing each of the

following values validating accuracy, validating loss ,training loss ,training accuracy If every value in training accuracy and validating accuracy has a high value, and every value in validating loss and training loss is low, it means that the model is well trained.

```
# Evaluating the model on the training, validating sets
score = model.evaluate(X_train, Y_train, verbose=0)
print("Training Accuracy: {1:.4f}, Training Loss: {0:.4f}".format(score[0], score[1]))

score = model.evaluate(X_valid, Y_valid, verbose=0)
print("Validating Accuracy: {1:.4f}, Validating Loss: {0:.4f}".format(score[0], score[1]))
```

```
Training Accuracy: 0.9997, Training Loss: 0.0009
Validating Accuracy: 0.9992, Validating Loss: 0.0021
```

Figure 5:Evaluating the model on the training, validating sets

```
import time
start = time.time()

# Compute the final loss and accuracy
final_res = model.evaluate(X_test,Y_test)
print("Testing final accuracy: {1:.4f}, Testing Final loss: {0:.4f}".format(final_res[0], final_res[1]))


end = time.time()
print ("Time elapsed inb seconds:", end - start)
```

```
174/174 [==============================] - 4s 22ms/step - loss: 0.0039 - accuracy: 0.9987
Testing final accuracy: 0.9987, Testing Final loss: 0.0039
Time elapsed inb seconds: 5.4411163330078125
```

Figure 6:Compute the final loss and accuracy

Table 2:Evaluating the model on the training, validating sets

Evaluating the model	**Training Accuracy**	**Training Loss**	**Validating Accuracy**	**Validating Loss**
Precent	**99,97%**	**o,o9%**	**99,92%**	**0,02%**

8. plot ROC curve

The ROC curve is a short form for Receiver Operating Characteristic curve. The performance of a classification model is represented by ROC curves. In terms of predicted probability, ROC tells us how good the model is at distinguishing between the given classes. The area under the curve (AUC) can be used as an indicator of the performance of the model.

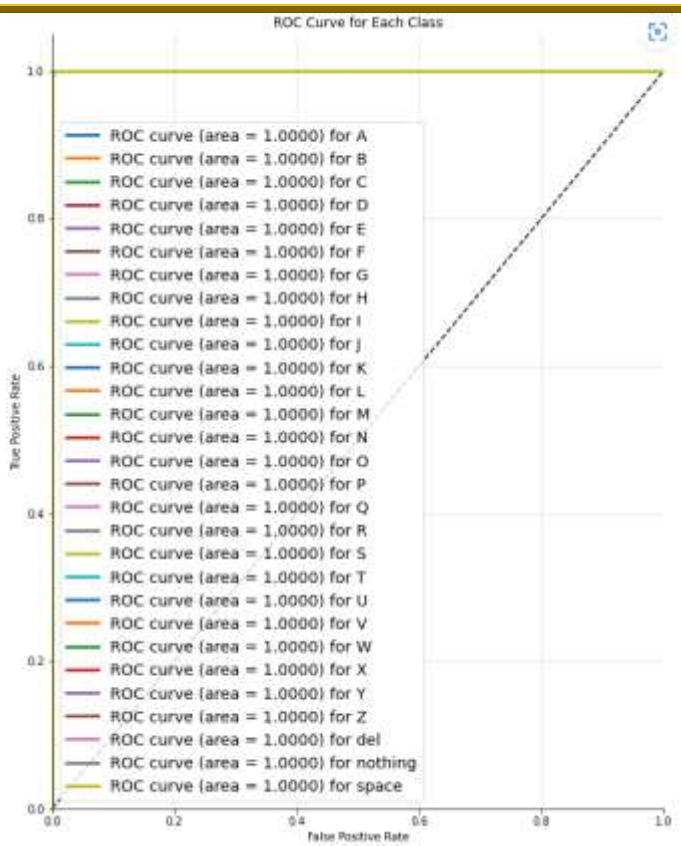

Figure 7: ROC curve

9. Classification Report

It is one of the metrics used to assess a classification-based machine learning model's performance. It shows the precision, recall, F1 score, and support of your model. It gives us a clearer picture of the overall effectiveness of our trained model. You must be familiar with all of the metrics shown in the classification report of a machine learning model. I have outlined all of the metrics below for your convenience so that you can quickly comprehend the categorization report generated by your machine learning model:

Table 3 :Classification Report defination

Metrics	Definition
Precision	Often known as the ratio of real positives to the total of true and false positives, is defined.
Recall	Which is the proportion of genuine positives to the total of true positives and false negatives.
F1 Score	The weighted harmonic mean of recall and precision is known as the F1. The projected performance of the model is higher the closer the F1 score value is near 1.0.
Support	The F1 is the weighted harmonic mean of recall and precision. The models predicted performance increases as the F1 score value gets closer to 1.0.

	precision	recall	f1-score	support
A	1.0000	1.0000	1.0000	183
B	1.0000	1.0000	1.0000	192
C	1.0000	1.0000	1.0000	187
D	1.0000	1.0000	1.0000	181
E	1.0000	1.0000	1.0000	182
F	1.0000	1.0000	1.0000	196
G	1.0000	1.0000	1.0000	186
H	1.0000	1.0000	1.0000	195
I	0.9947	1.0000	0.9973	187
J	0.9850	0.9945	0.9917	181
K	1.0000	1.0000	1.0000	189
L	1.0000	1.0000	1.0000	211
M	1.0000	1.0000	1.0000	188
N	1.0000	1.0000	1.0000	180
O	1.0000	1.0000	1.0000	212
P	1.0000	1.0000	1.0000	215
Q	1.0000	1.0000	1.0000	194
R	1.0000	1.0000	1.0000	196
S	1.0000	0.9950	0.9975	202
T	0.9945	1.0000	0.9972	181
U	1.0000	1.0000	1.0000	191
V	1.0000	0.9824	0.9911	170
W	0.9840	1.0000	0.9924	196
X	1.0000	1.0000	1.0000	203
Y	1.0000	0.9895	0.9947	191
Z	1.0000	1.0000	1.0000	181
del	1.0000	1.0000	1.0000	188
nothing	1.0000	1.0000	1.0000	191
space	1.0000	1.0000	1.0000	198
accuracy			0.9987	5547
macro avg	0.9987	0.9987	0.9987	5547
weighted avg	0.9988	0.9987	0.9987	5547

Figure 8:Classification Report

10. Confusion Matrix

It is a table that is utilized in categorization issues to determine where model errors occurred. The rows correspond to the actual courses for which the results were intended. The predictions we've made are represented by the columns. This table makes it simple to identify whose predictions were incorrect.

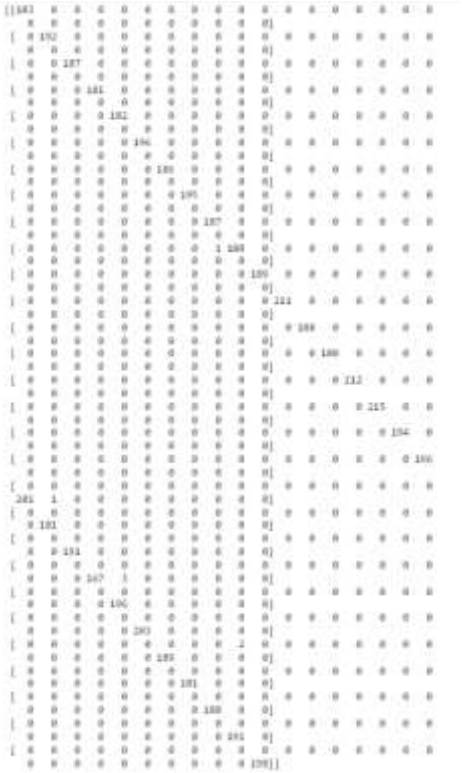

Figure 9: confusion matrix

5. CONCLUSION

In this paper we evaluated the performance of pretrained ResNet model of Deep Learning on datasets of 29 classes to signs we utilized pretrained model ResNet. we trained and validated the proposed model and tested its performance with un-seen dataset for testing. The Accuracy rate we achieved was 99.97% to 20 epoch. This indicates that our proposed model can effectively predicate and classify different sign language without error and with full performance but the VGG16 achieved higher accuracy 100% in pervious paper [1] and mobileNet achieved accuracy 95.41%. [2]

References:

1. Abu-Jamie, T.N. and S.S. Abu-Naser, Classification of Sign-language Using VGG16. International Journal of Academic Engineering Research (IJAER), 2022. 6(6).
2. Abu-Jamie, Tanseem N., and Samy S. Abu-Naser. "Classification of Sign-Language Using MobileNet-Deep Learning." International Journal of Academic Information Systems Research (IJAISR) 6.7 (2022).
3. Kumar, E.K., et al., Three-dimensional sign language recognition with angular velocity maps and connived feature resnet. IEEE Signal Processing Letters, 2018. 25(12): p. 1860-1864.
4. Nadgeri, S. and A. Kumar, An Analytical Study of Signs Used in Baby Sign Language Using Mobilenet Framework. Social Science Research Network, 2020.
5. Rezende, E., et al. Malicious software classification using transfer learning of resnet-50 deep neural network. in 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). 2017. IEEE.
6. Wang, M., et al., Scene classification of high-resolution remotely sensed image based on ResNet. Journal of Geovisualization and Spatial Analysis, 2019. 3(2): p. 1-9.
7. Liu, D., Y. Liu, and L. Dong. G-ResNet: Improved ResNet for brain tumor classification. in International Conference on Neural Information Processing. 2019. Springer.
8. Pu, J., W. Zhou, and H. Li, Dilated convolutional network with iterative optimization for continuous sign language recognition, in Proceedings of the 27th International Joint Conference on Artificial Intelligence. 2018, AAAI Press: Stockholm, Sweden. p. 885–891.
9. Daroya, R., D. Peralta, and P.C. Naval, Alphabet Sign Language Image Classification Using Deep Learning. TENCON 2018 - 2018 IEEE Region 10 Conference, 2018: p. 0646-0650.
10 Abdulhussein, A.A. and F.A. Raheem, Hand gesture recognition of static letters american sign language (ASL) using deep learning. Engineering and Technology Journal, 2020. 38(6): p. 926-937.
11. Yao, T., et al. Image recognition method of defective button battery base on improved MobileNetV1. in Chinese Conference on Image and Graphics Technologies. 2020. Springer.
11. Sinha, D. and M. El-Sharkawy. Thin mobilenet: An enhanced mobilenet architecture. in 2019 IEEE 10th annual ubiquitous computing, electronics & mobile communication conference (UEMCON). 2019. IEEE.
12.Sinha, D. and M. El-Sharkawy. Thin mobilenet: An enhanced mobilenet architecture. in 2019 IEEE 10th annual ubiquitous computing, electronics & mobile communication conference (UEMCON). 2019. IEEE.
13. Shinde, P.P. and S. Shah. A review of machine learning and deep learning applications. in 2018 Fourth international conference on computing communication control and automation (ICCUBEA). 2018. IEEE.
14. He, K., et al. Deep residual learning for image recognition. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.