Fingerspelling – A Sign Language Recognition

Research-based ☑ Application-based ☐

Name	Tan Jing Jie (Leader)	Ng Jan Hui	Sia Ken Yen
Programme	CS	CS	CS
ID	1804560	1802347	I804562
Contribution	1/3	1/3	1/3

1. INTRODUCTION

Sign language is a method of communication commonly used by deaf people or hearing-impaired people. Sign language is independent from the spoken language such as English, Chinese, and etcetera. [1] However, sign language has the limitation – not all things have a unique sign. Hence, the Fingerspelling has been introduced to draw support from spoken language – to spell out the names, places or things that do not have a unique sign. [2] Indeed, in physical daily life, there is a language gap for a sign language user and spoken language user.

The objectives of this project are "**To design a convolutional neural network (CNN) architecture with competitive performance.**", "**To investigate the relationship between reduction of color channels and model predictive performance.**", "**To investigate the effects of different regularization methods on model performance**.", and "**To investigate the performance of famous CNNs when being reimplemented.**"

In this paper, 3 different types of CNN architecture are focused and studied on the "**American Sign Language (Fingerspelling)**" (ASL) dataset. The chosen architectures are VGGNet16, ResNet18 and EfficientNetB0. Along with this, by reference to the Chapter 2 Related Work, two CNN architectures have been designed – **ShallowNet** and **Grayscale + CustomNet**. Additionally, several hypotheses were formulated in Chapter 4 Experiment & Evaluation as a direction to guide our research efforts.

2. RELATED WORK

First, the approach used by [3] for performing ASL recognition was done using a certain type of CNN model called VGG. The dataset was augmented to increase the dataset by 6 times to 14781 images. The authors specify the maximum ranges or degrees for shear, zoom, horizontal and vertical shifting in the augmentation process to prevent the classifier from overfitting. The images are resized to 224*224 pixels and scaled in the data preprocessing stage to ensure better performance in the training stage.

Furthermore, after thoroughly analyzing the paper [3], fine tuning hyperparameters is extremely vital. Besides, the paper concluded that learning rate and decay rate were the essential hyperparameters and initialized as 0.001 and 10^{-6} respectively. The goal of the former is to tune an optimal bound to eliminate the possibility of overfitting. The latter is to avoid exploding gradients during back propagation by adding a small penalty to the loss function. The achieved accuracy is 95.54%. Moreover, the result also discovered that the accuracy is deeply affected by the similar sign language such as the alphabet 'W' may always be misclassified with the alphabet 'O'. This indicates that the convolutional layer must be well-trained to avoid information loss before classification.

Besides, after studying the research papers that implement ResNet18 [4] and EfficientNet [5], it safely concluded that a global average pooling is vital as it is able to accept various types of input image dimension. This is a significant feature when the model is deployed for real-time use cases.

On the other hand, based on the works of [6] [7] [8], the authors employed self-designed Convolutional Neural Networks to perform classification on the American Sign Language (ASL) dataset gestures. The key takeaways from each paper are that shallow model architectures (at most 5 layers deep) were able to beat state-of-the-art CNNs when used to classify ASL gestures. The architecture of the models in the mentioned papers are similar in which the first few layers are organized into blocks of convolution layers followed by regularization methods of choice.

The authors used different regularization techniques to tackle overfitting issues. [6] used Dropout layers immediately after the double convolution layers of each block, and achieved 82.5% accuracy, while [7] [8] used Batch Normalization layers and got accuracies of 98.05% & 99.38% respectively. Apart from regularization, the three papers used image processing techniques that contributed to their high performance. [6] used image subtraction to segment the gesture from the background, while [8] binarized the input image before inputting to their neural network.

Based on the papers above, it is discovered that the configuration of using batch normalization layer after successive convolutional layers, followed by a single dropout layer before the end of the block, can reduce overfitting significantly whilst still maintaining high predictive performance. The concluded novelties are then implemented in Chapter 3 System Design that is discussed below.

3. SYSTEM DESIGN

In this research, the selected dataset is "American Sign Language (Fingerspelling)" dataset. This dataset consists of 87,000 images equally divided into 29 classes – "A" to "Z", "del", "nothing" and "space". The images are colored with pixel size 200*200. The top-down system block diagram is as follow.

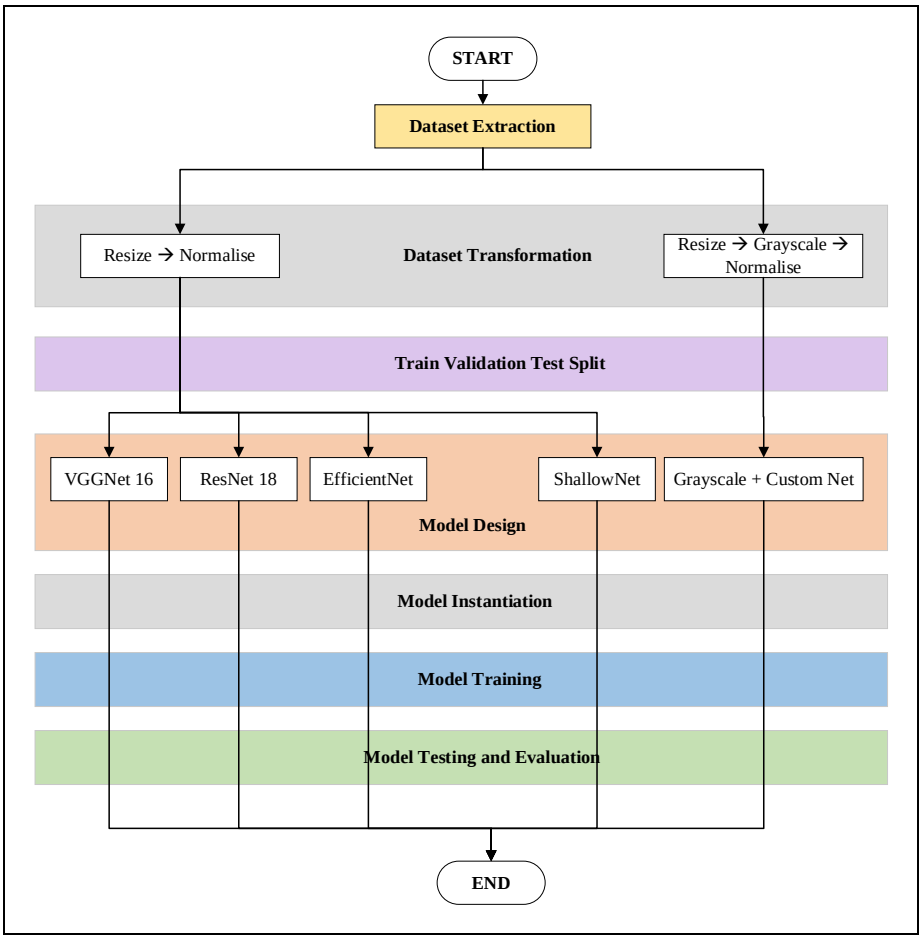


Figure 3.1 System Block Diagram

Firstly, the dataset is shrunken to only consist of 125 images per class to reduce the training time. By observation, it is noticed that the difference of neighbor images is scarcely – slight change of capture angle and having limited number (< 8 types) of background. Hence, during the dataset extraction, the algorithm has customized to produce higher variety of dataset.

Second, the extracted dataset has been loaded and transformed. At first, the image in the dataset is resized to suit for all CNN architecture. Next, the dataset is normalized by following the standard practice. Meanwhile, another additional transformation pipeline has added in middle – grayscale. This is to study the impact of color information in CNN. Afterward, the transformation data are spitted to training set, validation set and testing set.

The 3 well-known CNN architectures are implemented via transfer learning that are VGGNet, ResNet, and EfficientNet. For ResNet and EfficientNet, a custom FC layer with ReLU activation is concatenated on top layer. On the other hand, the nature of VGGNet contain high level feature so that it not required to explicit learn the high-level feature.

Next, ShallowNet, which is one of the custom designed models for this project that is inspired by the merits of [6] [7]. ShallowNet features 4 homogenous blocks of convolutional layers which are used for low-level feature extraction. Global Average Pooling was used to summaries the features of the convolution blocks into a 1D feature vector before passing into dense layers as this method was less prone to overfitting.

Another self-designed model called Grayscale + CustomNet inspired by [8], converts the input image to grayscale as a means of reducing the number of channels. This motive behind such image pre-processing is to further validate if the shallower layer is still working well on a simple dataset. Besides, this setup also aids in studying the impact of the number of convolutional layers towards the accuracy and inference time.

The table below tabulate the architectures of the proposed models.

Table 3.1 The architecture of ShallowNet and Grayscale + CustomNet

ShallowNet

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 222, 222]	1,792
ReLU-2	[-1, 64, 222, 222]	0
Conv2d-3	[-1, 64, 220, 220]	36,928
ReLU-4	[-1, 64, 220, 220]	0
BatchNorm2d-5	[-1, 64, 220, 220]	128
MaxPool2d-6	[-1, 64, 110, 110]	0
Dropout-7	[-1, 64, 110, 110]	0
Conv2d-8	[-1, 128, 108, 108]	73,856
ReLU-9	[-1, 128, 108, 108]	0
Conv2d-10	[-1, 128, 106, 106]	147,584
ReLU-11	[-1, 128, 106, 106]	0
BatchNorm2d-12	[-1, 128, 106, 106]	256
MaxPool2d-13	[-1, 128, 53, 53]	0
Dropout-14	[-1, 128, 53, 53]	0
Conv2d-15	[-1, 256, 51, 51]	295,168
ReLU-16	[-1, 256, 51, 51]	0
Conv2d-17	[-1, 256, 49, 49]	590,080
ReLU-18	[-1, 256, 49, 49]	0
BatchNorm2d-19	[-1, 256, 49, 49]	512
MaxPool2d-20	[-1, 256, 24, 24]	0
Dropout-21	[-1, 256, 24, 24]	0
Conv2d-22	[-1, 512, 22, 22]	1,180,160
ReLU-23	[-1, 512, 22, 22]	0
Conv2d-24	[-1, 512, 20, 20]	2,359,808
ReLU-25	[-1, 512, 20, 20]	0
BatchNorm2d-26	[-1, 512, 20, 20]	1,024
MaxPool2d-27	[-1, 512, 10, 10]	0
Dropout-28	[-1, 512, 10, 10]	0
AdaptiveAvgPool2d-29	[-1, 512, 1, 1]	0
Linear-30	[-1, 512]	262,656
Linear-31	[-1, 1024]	525,312
Linear-32	[-1, 29]	29,725

Grayscale + CustomNet

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 222, 222]	640
ReLU-2	[-1, 64, 222, 222]	0
Conv2d-3	[-1, 64, 220, 220]	36,928
ReLU-4	[-1, 64, 220, 220]	0
BatchNorm2d-5	[-1, 64, 220, 220]	128
MaxPool2d-6	[-1, 64, 110, 110]	0
Dropout-7	[-1, 64, 110, 110]	0
Conv2d-8	[-1, 128, 108, 108]	73,856
ReLU-9	[-1, 128, 108, 108]	0
Conv2d-10	[-1, 128, 106, 106]	147,584
ReLU-11	[-1, 128, 106, 106]	0
BatchNorm2d-12	[-1, 128, 106, 106]	256
MaxPool2d-13	[-1, 128, 53, 53]	0
Dropout-14	[-1, 128, 53, 53]	0
Conv2d-15	[-1, 256, 51, 51]	295,168
ReLU-16	[-1, 256, 51, 51]	0
Conv2d-17	[-1, 256, 49, 49]	590,080
ReLU-18	[-1, 256, 49, 49]	0
BatchNorm2d-19	[-1, 256, 49, 49]	512
MaxPool2d-20	[-1, 256, 24, 24]	0
Dropout-21	[-1, 256, 24, 24]	0
AdaptiveAvgPool2d-22	[-1, 256, 1, 1]	0
Linear-23	[-1, 256]	65,792
Linear-24	[-1, 512]	131,584
Linear-25	[-1, 29]	14,877

After the 5 CNN models have been designed, the process is moving on to the model instantization, training (using cross entropy as loss function), testing and evaluation.

4. EXPERIMENT & EVALUATION

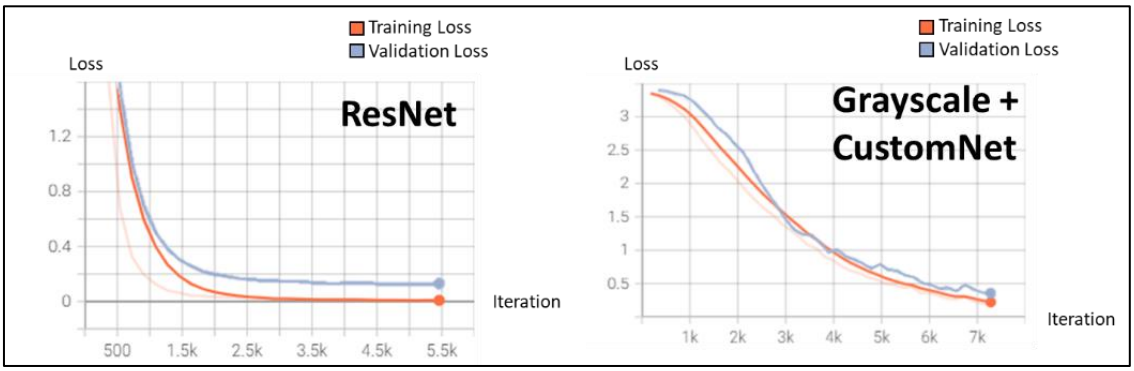
Experimentation to look for the optimal hyperparameters including fine tuning of epoch, learning rate, filter size, configuration and number of layers used, were conducted as part of our research efforts. The following table tabulates the fine tune direction in this project.

Table 4.1 The condition of loss value with respective fine-tuning actions taken

Condition	Fine tune action
The training loss and validation loss is decreasing, but training has ended.	Increase number of epochs
The training loss and validation loss are jittery	Decrease learning rate
The training loss is not decreasing	Decrease filter size

The graph below visualizes the training and validation loss of ResNet and Grayscale + CustomNet. This is to study the trainability of the proposed model. As shown, there is no overfitting issue as the validation loss is continuing to decrease (no rebound).

Figure 4.1 The visualization of loss value for ResNet and Grayscale + CustomNet



The table below shows the comparison of all the CNN models after 30 epochs. Easily observed that, ResNet and ShallowNet can be trained very fast. With a deep analysis, this project safely concluded that the higher the number of trainable layers, the higher the difficulty for a model to converse (need more epochs). However, shallowing the trainable layer (for not transfer learning model) incurs a shorter inference time.

Table 4.2 The comparison of different CNN architectures.

Architecture type	Transfer Learning			Proposed		
Model Net	**VGGNet**	**ResNet**	**EfficientNet**	**ShallowNet**	**Grayscale + CustomNet**	
Number of trainable layers	1	2	2	10	8	
Training Loss	0.6140	0.0071	0.4685	0.0593	Epoch=30 0.4391	Epoch=65 0.0271
Validation Loss	0.4350	0.0332	0.4638	0.1924	Epoch=30 0.4823	Epoch=65 0.0506
Inference time (s)	0.0104	0.0091	0.0489	0.0808	**0.0063**	

Based on the tabulated classification metrics for the top 3 performing models shown below, it is seen that our custom-built model – ShallowNet, was able to beat state-of-the-art models like EfficientNet and ResNet. ShallowNet was able to achieve 99.17% accuracy while Grayscale + CustomNet can achieve 98.07% accuracy. ShallowNet was able to generalize well to unseen samples due to the regularization techniques coupled with a large number of class specific features learned on the last few dense layers.

Table 4.3 The comparison of classification metrics for different CNN architecture.

ResNet18

Accuracy = 99.17%

	precision	recall	f1-score	support
A	1.00	1.00	1.00	13
B	1.00	1.00	1.00	14
C	1.00	1.00	1.00	14
D	1.00	1.00	1.00	12
E	1.00	1.00	1.00	14
F	1.00	1.00	1.00	10
G	1.00	1.00	1.00	11
H	1.00	1.00	1.00	8
I	0.94	1.00	0.97	17
J	1.00	1.00	1.00	11
K	1.00	1.00	1.00	17
L	1.00	1.00	1.00	12
M	1.00	1.00	1.00	11
N	1.00	1.00	1.00	11
O	1.00	1.00	1.00	16
P	0.86	1.00	0.92	6
Q	1.00	1.00	1.00	11
R	1.00	1.00	1.00	10
S	1.00	0.88	0.93	8
T	1.00	1.00	1.00	12
U	1.00	1.00	1.00	21
V	1.00	0.92	0.96	13
W	0.93	1.00	0.97	14
X	1.00	1.00	1.00	8
Y	1.00	1.00	1.00	16
Z	1.00	1.00	1.00	15
del	1.00	1.00	1.00	12
nothing	1.00	1.00	1.00	9
space	1.00	0.94	0.97	16
accuracy			0.99	362
macro avg	0.99	0.99	0.99	362
weighted avg	0.99	0.99	0.99	362

ShallowNet

Accuracy = 99.17%

	precision	recall	f1-score	support
A	1.00	1.00	1.00	12
B	1.00	1.00	1.00	14
C	1.00	1.00	1.00	8
D	1.00	1.00	1.00	13
E	1.00	1.00	1.00	10
F	1.00	1.00	1.00	10
G	1.00	1.00	1.00	7
H	1.00	1.00	1.00	17
I	1.00	1.00	1.00	16
J	1.00	1.00	1.00	13
K	1.00	1.00	1.00	12
L	1.00	1.00	1.00	11
M	1.00	0.93	0.96	14
N	0.92	1.00	0.96	12
O	0.93	1.00	0.96	13
P	1.00	1.00	1.00	10
Q	1.00	0.94	0.97	18
R	1.00	0.92	0.96	13
S	1.00	1.00	1.00	12
T	1.00	1.00	1.00	8
U	0.92	1.00	0.96	11
V	1.00	1.00	1.00	13
W	1.00	1.00	1.00	12
X	1.00	1.00	1.00	10
Y	1.00	1.00	1.00	16
Z	1.00	1.00	1.00	17
del	1.00	1.00	1.00	15
nothing	1.00	1.00	1.00	16
space	1.00	1.00	1.00	9
accuracy			0.99	362
macro avg	0.99	0.99	0.99	362
weighted avg	0.99	0.99	0.99	362

Grayscale + CustomNet

Accuracy = 98.07%

	precision	recall	f1-score	support
A	1.00	1.00	1.00	10
B	1.00	1.00	1.00	12
C	1.00	1.00	1.00	15
D	1.00	1.00	1.00	8
E	1.00	1.00	1.00	12
F	1.00	1.00	1.00	9
G	1.00	0.92	0.96	12
H	1.00	0.93	0.96	14
I	1.00	1.00	1.00	15
J	1.00	1.00	1.00	15
K	0.93	1.00	0.97	14
L	1.00	1.00	1.00	15
M	0.92	1.00	0.96	12
N	1.00	0.91	0.95	11
O	1.00	1.00	1.00	15
P	0.94	0.94	0.94	18
Q	0.88	1.00	0.93	7
R	1.00	1.00	1.00	15
S	1.00	1.00	1.00	11
T	1.00	1.00	1.00	15
U	0.93	1.00	0.97	14
V	0.92	0.92	0.92	13
W	1.00	1.00	1.00	7
X	1.00	0.93	0.96	14
Y	1.00	1.00	1.00	14
Z	0.90	1.00	0.95	9
del	1.00	1.00	1.00	11
nothing	1.00	1.00	1.00	9
space	1.00	0.94	0.97	16
accuracy			0.98	362
macro avg	0.98	0.98	0.98	362
weighted avg	0.98	0.98	0.98	362

Refer to this link to view the classification metrics and confusion matrix for each of the models. Upon review, as tabulated in table below, the self-proposed model in this research achieves better performance compared to other existing models using ASL datasets.

Table 4.4 The comparison and benchmarking with other works.

Authors	**Model**	**Accuracy**	**Inference time**
Garcia and Viesca, 2016 [9]	GoogLeNet transfer learning	70%	-
Kania, K.et. al., 2018 [10]	Wide Residual Networks transfer learning	93.30%	-
Masood, S.et. al., 2018 [3]	VGG16 transfer learning	94.68%	0.0104 seconds
Our work	**ShallowNet**	**99.17%**	**0.0808 seconds**
	Grayscale + CustomNet	**98.07%**	**0.0063 seconds**

During performing experiments and simulations on various models, several challenges were encountered. Initially, the VGG16 architecture was reimplemented to investigate whether that model initialization had significant impacts on its performance. It was observed that the training loss for every epoch had little to no difference, which indicated the model was not learning and fitted very badly on the dataset. This led to the establishment of the first hypothesis of the research – **"Reimplementation of prominent CNN models affects the training process due to the improper initialization of weights."** The hypothesis was tested by experimenting the recreated VGG model by training on the much popular CIFAR10 dataset. The outcome is CIFAR10 dataset did show that the model was loss was in fact decreasing, and this has led us to reject the first hypothesis.

The second hypothesis that was raised is, **"Training on images with lesser than 3 channels, would cause the model to learn non-representative features and perform badly."**. This hypothesis was in line with our second objective. This hypothesis was tested by implementing the Grayscale + CustomNet, which will convert the input images to grayscale before learning its features. Converting an image to grayscale, effectively shrinks the number of channels to 1. However, the predictive performance of our Grayscale + CustomNet was able to outperform the pretrained models. This led to the rejection of the second hypothesis.

With knowledge of previous experiments, it is known that models built from scratch requires longer training times and tend to overfit due to having many trainable parameters, as opposed to their transfer learning counterparts. Hence, the third hypothesis was raised, **"Models that are built from scratch are prone to overfitting"**. A third experiment was conducted by training the models built from scratch and observing its performance. The CustomNet and ShallowNet models were put to the test. The results shows that although ShallowNet and CustomNet had much more parameters to train compared to pretrained models, the training times were at par with the pretrained models, and still possessed better predictive performance. The results had led us to reject this hypothesis due to our efforts of finding the balance between optimal configuration of layers and regularization methods.

Another hypothesis that was raised is, **"Models that are built from scratch may outperform pretrained models on simple datasets.".** After fine-tuning, both the Grayscale + CustomNet and ShallowNet able to beat the pretrained networks. On this basis, these lightweight models having a less inference time. This hypothesis was in line with our objective of designing a CNN with competitive performance, hence, this hypothesis was accepted.

5. CONCLUSION

In conclusion, the objectives were achieved. By using the merits from well-known architectures, this project had successfully developed 2 novel models – ShallowNet (accept colored image) and CustomNet (accept grayscale image).

The performances of our self-designed CNNs can compete and even outperform certain CNN architectures found in related works. A conclusion can be drawn which is pretrained and deeper models are able to train with a shorter time, but custom-built models that are usually less complex and shallow, and are tailored to the datasets were able to generalize and perform better. This is a tradeoff that can be considered depending on the model performance required. The future work for this research would be directed on reducing training time of self-designed models by tuning the initialization of model parameter.

6. CONTRIBUTIONS

- In this report, the contributions of each member are equal, while in coding, the contribution is indicated by the code tag.
- The Google Drive folder link (with Google Collab Notebook): https://drive.google.com/drive/folders/1-yB27djKv71A2JKVl_PjFQrcxhOxyhZ6?usp=sharing
- Dataset: https://www.kaggle.com/datasets/grassknoted/asl-alphabet

REFRENCES

[1] J. Cummins, "The Relationship between American Sign Language Proficiency and English," 2010.

[2] B. S. "Fingerspelling Alphabet," British Sign, 2022. [Online]. Available: https://www.british-sign.co.uk/fingerspelling-alphabet-charts/. [Accessed 13 April 2022].

[3] S. Masood, H. C. Thuwal and A. Srivastava, "American Sign Language Character Recognition Using Convolution Neural Network," in *Smart Computing and Informatics*, 2018, pp. 403-412.

[4] H.-w. Zhang, Y. Hu, Y. J. Zou and C. Y. Wu, "Fingerspelling Identification for American Sign Language Based on Resnet-18," *International Journal of Advanced Networking and Applications (IJANA)*, vol. 13, no. 1, pp. 4816-4820, 2021.

[5] N. Gupta, Gesture Detection using Tensor flow Lite EfficientNet Model for Communication and E-learning Module for Mute and Deaf, vol. 10, India, 2022, p. 13.

[6] V. Bheda and D. Radpour, "Using Deep Convolutional Networks for Gesture Recognition in American Sign Language," arXiv preprint arXiv:1710.06836, 2017.

[7] P. Das, T. Ahmed and M. F. Ali, "Static Hand Gesture Recognition for American Sign Language using Deep Convolutional Neural Netork," in *2020 IEEE Region 10 Symposium (TENSYMP)*, Dhaka, 2020.

[8] A. Kasapbaşi, A. E. Ahmed Elbushra, O. Al-hardanee and A. Yilmaz, "DeepASLR: A CNN based human computer interface for American Sign Language recognition for hearing-impaired individuals," *Computer Methods and Programs in Biomedicine Update*, vol. 2, 2022.

[9] B. Garcia and S. A. Viesca, "Real-time American Sign Language Recognition with Convolutional Neural," *Convolutional Neural Networks for Visual Recognition*, pp. 225-232, 2016.

[10] K. Kania and U. M. Kaczmar, "American Sign Language FingerspellingRecognition Using Wide Residual Networks," in *Artificial Intelligence and Soft Computing*, Springer, 2018, pp. 97-107.