Generalization of Bangla Sign Language Recognition using Angular Loss Functions

SAMIYA KABIR YOUME, TOWSIF ALAM CHOWDHURY, HOSSAIN AHAMED, MD. SAYEED ABID, LABIB CHOWDHURY, and NABEEL MOHAMMED

Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh

Corresponding author: Nabeel Mohammed (e-mail: nabeel.mohammed@northsouth.edu)

ABSTRACT Sign Language provides the means of conveying messages for deaf and mute people. Effective communication with the masses is a great challenge for the deaf and mute community, as Sign Language is not commonly understood. Many researchers have done numerous works in foreign language datasets like English, French, Japanese, etc. However, for Bangla, one of the most widely spoken languages, much significant work has not been done yet. Most of the works on Bangla Sign Language are executed on small datasets and report satisfactory performance. However, when small datasets are evaluated from the perspective of generalizability, particularly when using deep learning based solutions, these models fail to reproduce to the same performance. Therefore, this paper poses inter-dataset evaluation as the main evaluation criteria and evaluates several deep learning based models. This evaluation is done for Bangla by leveraging two popular datasets of Bangla Sign Language. Unsurprisingly, the inter-dataset performance is inferior, and several approaches to improve are explored and documented, including the use of angular margin based loss functions. The results demonstrate the importance of such an evaluation and also show that one of the proposed approaches shows promising performance, albeit with significant room for improvement. This raises the need for a standardized dataset to overcome this issue of generalization for real-life applications, as well as the need to encourage future works to concentrate on challenging evaluations instead of pursuing deceptively good intra-dataset performance.

INDEX TERMS Angular loss functions, Bangla Sign Language, Convolution neural network, Deep learning, Finger-spelling, Generalization

I. INTRODUCTION

LANGUAGE is the core tool used in expressing one’s thoughts, ideas, and increasingly social identity as it plays an inevitable role in the simple yet powerful act of passing a message to some other. Nevertheless, some individuals are deprived of the ability to speak and hear. These individuals may not be able to communicate through common means, yet they most definitely need to communicate. The dominant approach to affect communication is the use of a special gesture-based method, popularly called Sign Language. Around 3 million hearing-impaired people are living in Bangladesh [1]. Deaf and mute people face major difficulty communicating with normal healthy people, especially in Bangladesh with lower social awareness. Such situations require the use of interpretation of Sign Language, as very few people can comprehend it. Currently, the only way of communication is via human interpreters. Still, the availability of such expertise is scarce, which becomes a significant barrier in the effective integration of the deaf and mute communities. As a result, due to lack of expressiveness, they face critical challenges in education, socialization, medical therapy, and significantly in gaining employment. Therefore, automatic Sign Language recognition can play a vital role in more effective interaction with the special needs community to bridge the gap. Over the past few decades, image-based machine learning, and deep learning techniques, have stood out in the field of recognition. These techniques are then utilized to translate hand signs in text or audio form so that the hearing majority can understand easily [2].

The two dominant approaches to Sign Language recognition are based on inputs, namely vision-based and sensor-based methods [2]. Vision-based techniques are conducted by
obtaining images from a camera without any use of extra sensors or special gloves, whereas, in the sensor-based methods, gloves are worn by the signer, or sensors are used to recognize the signs. Although it may be reasonably inconvenient to carry and use these sensors for practical purposes, these devices are not always available or affordable. Generally, gestures are classified into static and dynamic forms in every Sign Language, where static signs are still image forms, and dynamic signs are represented by video capturing motion. Sign Language recognition can be categorized into three major components: finger-spelling, word-level sign vocabulary, and non-manual features as in facial expressions [3]. Most of the researchers mainly worked on the finger-spelling type dataset of Sign Language. In this category, signs are divided into further sub-categories depending on the number of hands used (one-handed and two-handed methods).

There is no widely adopted international Sign Language, and for every country, Sign Languages vary from region to region. Standard benchmark Sign Language datasets are available for high resource languages, such as English, Arabic, Chinese, etc. Shi et al. [4] introduced the largest American Sign Language (ASL) dataset using naturally occurring video data. Kang et al. [5] made a dataset consisting of 1,000 images for each of the 31 different hand signs from five subjects, including alphabets and numbers for ASL recognition. A dataset consisting of over 500 samples of each symbol, which makes a total of 48,000 samples, were recorded from 4 different persons by Pugeault et al. [6] to build a large-size ASL dataset. Rioux-Maldague et al. [7], Aryanie et al. [8], and Ameen et al. [9] used this dataset introduced by Pugeault et al. for the classification of ASL. Hu et al. [10], in their work, used a large dataset of 120,000 images representing 24 alphabet letters. Apart from ASL, Nakjai et al. [11] made a dataset for the Thai Sign Language classification, where data was collected from 11 signers. Parvat et al. [12] also made a similar Thai dataset from five hand signers. Jiang et al. [13] developed a self-collected dataset containing 1320 images with the help of 44 volunteers, which were later augmented for Chinese Sign Language. For Arabic Sign Language, a dataset has been developed by Aly et al. [14], where the 28 Arabic alphabets were repeated 50 times by three different users. This dataset has more samples than the one made by Dahmani et al. [15], where the authors made a dataset of 30 hand signs from 30 different subjects who repeated the 30 letters on an average of 4 times. Similarly, there exist well-enriched and well-established datasets for other popular languages. However, for Bangla Sign Language (BSL), there are no large datasets that can take full advantage of machine learning and deep learning methods for recognition despite being the seventh most popular language with 265 million speakers [16].

Most of the works that have been done so far on Bangla Sign Language (BSL) have relied on small author-curated datasets. Among these datasets, the Ishara-Lipi dataset introduced by Sanzidul et al. [17] is commonly utilized to identify Bangla hand gestures, as it is pre-processed and publicly available. The collection consists of 2800 samples, of which 1000 are for digits, and the rest 1800 are for alphabets. This amount of data is inadequate for robust and effective deep learning models. Shafigul et al. [18] constructed a dataset for BSL containing 30916 samples, of which 23864 are alphabets and 7052 are digits. This is the extensive dataset on BSL that we could find when we initiated our study. While there are numerous papers reporting works on BSL, the datasets are either too small or not publicly available.

This paper demonstrates that although the studies on automatic BSL recognition may seem promising, the results are only impressive on individual collections. The exact solutions do not generalize well, as shown in this study using inter-dataset evaluation. The proposed study aims to investigate the current state-of-the-art BSL recognition and present the necessity of generalization of BSL recognition in order to deploy models for real-world scenarios where the data can be more diverse. This study conducted multiple experiments in both inter-dataset and intra-dataset testing schemes in order to evaluate the existing hypothesis with more diverse data. Although the intra-dataset test performance is promising, inter-dataset test performance raises a question of the importance of generalization capability that utilizes several recent architectures and loss functions. Our findings demonstrate that, while the intra-dataset performance of the model is satisfactory, the inter-dataset performance is comparably low. In order to improve the inter-dataset performance, we have experimented with different angular margin loss functions, as angular margin loss functions have shown some success in better generalization in face recognition [19] and speaker recognition [20] in the last few years. After incorporating these specialist loss functions, results present relatively better inter-dataset performance compared to other more widely used methods. However, the improvement does not result in a practically usable outcome because the two datasets’ samples of corresponding classes are quite different from each other in terms of angle, light contrast, orientation, hand size, and background. This paper also presents an investigation on biases in training samples using visualization of Grad-CAM. The findings of this study can serve as an inspiration for research on both finding more generalized methods and making a more diverse dataset on BSL recognition. The contributions of this paper are summarized as follows:

- To the best of our knowledge, this is the first study that evaluates automatic finger-spelled BSL recognition in an inter-dataset setting. Our findings of poor performance will open new evaluation methods and lead to better-generalized solutions.
- We have performed a thorough investigation using the VGG19 model and demonstrated that using a specific intermediate feature map results in superior intra and inter dataset performance.
- We have explored the use of multiple angular margin-based functions. In one case, the use of the Sphereface loss improves inter-dataset performance by 8%.
The rest of the paper is organized as follows: the following section summarizes the existing literature and present the different loss functions which are used in our work, section 3 describes the datasets we have used, data processing, and methodology of this work, the different experimental configurations and experimental details are described in section 4. Section 5 presents results obtained from the different configurations followed by their discussion in section 6. An ablation study is presented in section 7 and section 8 concludes the paper.

II. BACKGROUND STUDY

This section discusses various researches implemented on Sign Language recognition. Many studies employ machine learning or deep learning techniques, and some even incorporate image processing to achieve a better outcome. The last part arranges a brief analysis of different loss functions, particularly those applied in this work.

A. MACHINE LEARNING TECHNIQUES

Before deep learning, machine learning algorithms were implemented effectively for recognizing static images. Aryanie et al. [9] proposed a machine learning approach that recognizes finger-spelling ASL characters using K-Nearest Neighbour (KNN) algorithm along with Principal Component Analysis (PCA) for dimensionality reduction. Mukai et al. [21] presented in their research a Japanese finger-spelling recognition system using classification tree and Support Vector Machine (SVM) and achieved 86% accuracy. Parwat et al. [12] also implemented SVM on Thai finger-spelling recognition. They extracted features from input images locally as well as globally and showed comparisons on different types of SVM kernels. In the specific context of BSL, Hasan et al. [22], in their work, proposed a Linear Support Vector Machine (LSVM) for classification on a dataset of 5400 images for training. Compared to the KNN algorithm, which gave 90.94% accuracy, their model accuracy was 96.463%. The proposed system provides a low-cost translator that can convert an image to text. Hasan et al. [23] also used the SVM algorithm as a classifier for their work after extracting HOG descriptors. A dataset of 320 sample images was used, where 64 images were used for testing, and the dataset was divided into nine classes, with each category containing 20 samples. This small amount of data is inadequate for robust training and evaluation. Similar to the work of Hasan et al. [23], Uddin et al. [24] also trained an SVM on a dataset of 2400 images, which was then evaluated on another 2400. They reported an accuracy of 97.7% with their method, which involved converting RGB images to the HSV color space, extracting features using a bank of Gabor Filters, and using Kernel PCA for dimensionality reduction. Yasar et al. [25] in their work, trained on a relatively small dataset of 330 RGB images using PCA and Linear Discriminant Analysis (LDA) to minimize intra-class scatter and maximize inter-class distance. Yasar et al. [26] used SIFT features on a vocabulary dataset of only 90 images. KNN and SVM were then trained on these features. A recurring theme in these studies and the self-collection of small datasets, most of which are not available for comparative study.

B. DEEP LEARNING TECHNIQUES

Image-based deep learning techniques have been used in many works for Sign Language classification. Aly et al. [14] introduced an Arabic Sign Language recognition system using PCANet, which helps to identify local features from the depth and intensity of images. Ameen et al. [8] developed a convolutional neural network (CNN) for classifying finger-spelling images and reported precision of 82%, and recall of 80%. Hu et al. [10] presented Deep Belief Network (DBN) with three Restricted Boltzman Machines (RBMs) for classifying ASL. Maldague et al. [7] also introduced DBN for ASL recognition and achieved 99% recall. This work focused on improving real-time ASL recognition while being robust to different light intensities. Kang et al. [5] also applied a CNN architecture to classify finger-spelling in ASL while also incorporating information about the position of the thumb. Nakajai et al. [11] combined two approaches: CNN and Histogram of Oriented Gradients (HOG), and they reported 91.26% of precision for Thai finger-spelling recognition.

Most of the works for BSL used CNN-based architecture along with softmax loss for the classification of images. Shafiqul et al. [18], made the extensive dataset for BSL, consisting of 30916 samples which were collected from 25 different students. They developed a CNN-based architecture and reported an accuracy of 99.83%, 100%, and 99.80%, respectively, for basic characters and numerals and combined usage. Ahamed et al. [27] used a dataset of 518 image samples in an artificial neural network (ANN) while identifying fingertip position for recognition and achieved a recognition rate of 99%. A dataset consisting of 1000 gestures and 1000 letters was made by Hoque et al. [28] and was used to train a Region-based Convolutional Network. Their model achieved an accuracy of 98.20% on their dataset. Rafi et al. [29] gathered a dataset of 12581 images of hand signs for 38 alphabets. They proposed a network of VGG19 architecture to recognize static hand gestures with an accuracy of 89.6%. Similarly, Urmeet et al. [30] employed the Xception architecture on a dataset of 2000 images with 37 different signs obtaining an accuracy of 98.93%. Yasar et al. used a virtual reality-based hand tracking controller to track the hand motion using CNN-based architecture. They operated this to generate a limited amount of data for their work with an error rate of 2% [31].

Sanzidul et al. [32] introduced Ishara-Bochon in their work, a processed dataset of BSL digits. This dataset consists of 1000 images of Bangla sign digits from 0 to 9, and images were fed into a CNN model while obtaining an accuracy of 92.87%. Sanzidul et al. [33] also built the Ishara-Lipi dataset with 1800 image samples, where they worked on only with alphabets of 36 classes of the BSL. Sanzidul et al. [34] proposed another study that employed this dataset and train a CNN architecture, reporting accuracy of 92.74% and...
94.74% on the training and validation sets, respectively. Both the Ishara-Bochon and the Ishara-Lipi datasets are made open source for public contribution. In another work, Sanzidul et al. used the Bangla Sign numerals dataset (Ishara-Bochon), which underwent image processing wherein the images were converted into grayscale. Those images were then classified using a CNN architecture and reported an accuracy of 95% [35]. This numeral dataset was also utilized by Hossain et al. [36] in their work, and a capsule network has been used for recognizing digits of BSL. Their proposed model achieved better performance than the previous work done by Sanzidul et al. [35] with an accuracy of 98.84%. The Ishara-Lipi dataset was again used by Hasan et al. [37] and achieved an accuracy of 99.22%. Nihal et al. [38] presented a robust model by establishing a standard automated recognition system for seen and unseen data utilizing two approaches: transfer learning and zero-shot learning (ZSL). Furthermore, they contributed to making a BSL dataset by gathering nine different datasets with a variation in the background, hand size, capture angle, skin tone. They trained 378 models on a small dataset and found that, out of 378 models, Densenet201 followed by LDA classifier has the best performance. Thus, they trained their best model on a large dataset and reported 93.68% accuracy. Most of the researchers obtained their proposed system using VGG16 architecture with some changes in the classification layer. Their system performed well with their respective dataset. However, interestingly we have not found any studies that examine the generalizability of the solutions. We aim to fill this gap by performing an inter-dataset evaluation.

C. LOSS FUNCTIONS

Our work mainly focuses on the generalization of solutions so that the Sign Language systems can perform robustly. The prior works for recognizing BSL used Deep Learning models with the traditional softmax loss. In this work, we have experimented with different loss functions to get preferable results for obtaining better performance in inter-dataset evaluations. Experiments are done with SphereFace, CosFace, and ArcFace, as these loss functions have achieved noticeable results in face recognition and have produced robust models in that domain [39] [40] [19]. These angular loss functions have usually been used in face recognition tasks, and very few researchers used these loss functions for other recognition tasks. Chowdhury et al. [20] proposed in their work a joint angular margin loss for speaker recognition and achieved a notable performance improvement in an inter-dataset evaluation setting. This work aims to analyze the generalizability of some deep learning hand sign recognition models when trained on the previously mentioned publicly available datasets. As angular margin losses have proven to be useful in building better-generalized face recognition models, we have used them in this context of BSL recognition.

III. METHODOLOGY

A. DATASETS

| Bangla Alphabets | English Transliteration |
|-----------------|------------------------|
| অ - a           | A                       |
| আ - a           | A                       |
| ই - i           | I                       |
| এ - e           | E                       |
| ও - o           | O                       |
| ক - ka          | K                       |
| খ - kha         | Kh                      |
| গ - ga          | G                       |
| ঘ - gha         | Gh                      |
| চ - ca          | C                       |
| ছ - cha         | Ch                      |
| জ - ja          | J                       |
| ঝ - jha         | Jh                      |
| ঞ - ta          | T                        |
| ট - tha         | Tha                     |
| ঠ - da          | Da                      |
| ড - dha         | Dh                      |
| ণ - na          | Na                      |
| ত - pa          | Pa                      |
| থ - bha         | Bha                     |
| দ - ma          | Ma                      |
| ধ - ya          | Ya                      |
| ধ - ra          | Ra                      |
| ণ - la          | L                        |
| ঔ - na          | Na                      |
| ক - ka          | K                       |
| খ - kha         | Kh                      |
| গ - ga          | G                       |
| ঘ - gha         | Gh                      |
| চ - ca          | C                       |
| ছ - cha         | Ch                      |
| জ - ja          | J                       |
| ঝ - jha         | Jh                      |
| ঞ - ta          | T                        |
| ট - tha         | Tha                     |
| ঠ - da          | Da                      |
| ড - dha         | Dh                      |
| ণ - na          | Na                      |
| ত - pa          | Pa                      |
| থ - bha         | Bha                     |
| দ - ma          | Ma                      |
| ধ - ya          | Ya                      |
| ধ - ra          | Ra                      |

BSL consists of 10 gestures for digits, 38 gestures for alphabets, and about 4000 double hand symbols for commonly used Bangla words [41]. The experiments performed for this study were conducted on two different datasets, with an aim to test the inter dataset performance, as a proxy for generalization of the models. These are the "Ishara-Lipi dataset" [17] and the dataset used by Shafiqul et al. [18] which is referred in our paper as the "BdSL dataset". From both datasets, only images of BSL for Bangla Characters were used in the experiments. All the images in both datasets are in .jpg format, having a clear background without any noise. The datasets are explained in more detail below.

FIGURE 1: Gestures of Bangla Sign Language
horizontally, and the brightness level is also enhanced.

C. PROBLEM FORMULATION

We formulate our evaluation criteria by assuming, the training set and test set of BdSL dataset as, $X_{BdSL-Train} = \{x_1, x_2, x_3, ..., x_N\}$, and $X_{BdSL-Test} = \{x_1, x_2, x_3, ..., x_N\}$, respectively. We also consider training set and test for Ishara-Lipi dataset as, $X_{Ishara-Train} = \{x_1, x_2, x_3, ..., x_N\}$, and $X_{Ishara-Test} = \{x_1, x_2, x_3, ..., x_N\}$, respectively, where each set has $N$ number of signed images of 35 classes. We also assume label as $Y_c$, where $c \in [1, 35]$. The goal of our research is to compare the performance of different deep learning models on intra and inter datasets. So, for intra dataset evaluation, we have trained and tested the model on the same dataset; for instance, when deep learning models are trained on $X_{BdSL-Train}$, then $X_{BdSL-Test}$ images are given for testing the model. On the other hand, we can assess inter-dataset performance by training models on one dataset and testing on another. In particular, for a given test set of $X_{BdSL-Test}$ sign images, we aim to assign a label $Y_c$ while the model is trained on $X_{Ishara-Train}$.

D. MODEL

Recent deep learning approaches, specifically CNN, have outperformed prior state-of-the-art machine learning techniques, particularly in computer vision tasks. Most commonly, CNN is used to analyze visual imagery to recognize/classify an object. This study focuses on using such deep learning architectures to classify the type of Bangla finger-spelled signs presented as static images in figures 2 and 3. This work considered several modified pre-trained CNN architectures, including AlexNet [43], ResNet50 [44], VGG16, VGG16 with batch normalisation (VGG16-BN), VGG19 and VGG19 with batch normalization (VGG19-BN) for transfer learning [45]. To speed up the learning process and improve the generalizability of the network [46], we use pre-trained models to train our pre-processed datasets. We have discarded the classification layer from the pre-trained models and experimented with different combinations of fully connected layers after the final feature vector to determine the optimal number of layers and the number of neurons in each layer after the feature vector. Our experiments discovered that the combination of 512-256 after the final feature vector showed the best results. Between the fully connected layers, batch normalization layers are also integrated to stabilize the learning process [47]. For model selection, we have used a softmax function in the last fully connected layer. Rectified linear units (ReLU) activation function is used for all of the models, as it provides non-linearity to the network that helps in the improvement in generalization of the classifier by saturating the output [48] [49]. Later, we have continued our further experiments with angular losses using the model that produced the best results among the six networks.
The mathematical expression of softmax loss defined as scores, and then projects them to a probability distribution. The softmax loss combines softmax function and cross-entropy loss. 1) Softmax Loss

The softmax loss combines softmax function and cross-entropy loss [40]. Softmax function processes the output of a fully connected layer, whose values are usually interpreted as scores, and then projects them to a probability distribution. The mathematical expression of softmax loss defined as equation (1)

\[
L_{\text{softmax}} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{W_{yi}^T x_i + b_{yi}}}{\sum_{c=1}^{C} e^{W_{yi}^T x_i + b_{yc}}}
\]

where \(x_i\) represents the learned feature vector of \(i\)-th sample corresponding to the \(y_i\)-th class. \(W\) denotes the weight matrix of the last fully connected layer (\(c \in [1, C]\), \(C\) is the number of classes) and \(N\) is the number of batch size. Here, we fix bias as 0 for simplicity.

2) SphereFace Loss

Angular Softmax loss function is established from the traditional softmax loss function by manipulating decision boundaries to construct angular margin. The inner dot product of input feature vectors and weights from equation (1) can be expressed as,

\[
W_{c}^T x_i = ||W_{c}|| \|x_i\| \cos(\theta)
\]

where \(\theta_{c}\) is the angle between \(W_{c}\) and \(x_i\). From equation (2), it is evident that the posterior probability depends on both norm and angle vectors. As a result, normalizing the weight vector by \(L2\) normalization, the cosine angle only contributes to the prediction. The size of the angular margin can be adjusted by a positive parameter \(m\) for the basis of learning an angular margin between different distinctive classes. The proposed loss of [39] is formulated as,

\[
L_{\text{SphereFace}} = \frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{\psi(\theta_{yi,i})}}{e^{\psi(\theta_{yi,i})} + \sum_{j \neq y_i} e^{\psi(\theta_{yi,j})}}
\]

subject to

\[
\psi(\theta_{yi,i}) = (-1)^k \cos(m \theta_{yi,i}) - 2k
\]

\[
\theta_{yi,i} \in \left[\frac{k \pi}{m}, \frac{(k + 1) \pi}{m}\right]
\]

where \(k \in [0, m - 1]\)

3) Large Margin Cosine Loss

Similar to SphereFace [39], the author [40] has also normalized weights by \(L2\) normalization along with embedding vector, which is set as scaling parameter, \(s\). The scaling parameter adjusts the magnitude of radius on the hypersphere manifold. To maximize the decision margin, they introduced hyper-parameter, \(m\) in the cosine space rather than angular space [40]. The Large Margin Cosine loss is defined as follows,

\[
L_{\text{CosFace}} = -\frac{1}{N} \sum_{i=1}^{N} \log e^{\cos(\theta_{yi,i}) - m} + \sum_{c \neq y_i} e^{\cos(\theta_{cy,i}) - m}
\]

where

\[
\cos(\theta_{c,i}) = W_{c}^T x_i
\]

where \(\theta_{c}\) is the angle between \(x_i\) which is the \(i\)-th feature vector of corresponding ground truth \(y_i\) and \(W_{c}\) is the weight of \(c\)-th class and \(N\) is the number of batch size. In the [40], they exemplify that the discriminative capability of the learned features can be increased by raising the margin \(m\).
4) ArcFace Loss

In this work, the authors [19] have also normalized both the deep feature vector and weight vector to project the embedding vectors on hypersphere with a scaling constant $s$. They have introduced additive angular margin $m$ to the target angle in the angular space to simultaneously enhance the concentration in intra-class and divergence between inter-class data [19]. Besides, this study also illustrates that ArcFace is better in comparison with the other loss functions SphereFace, and CosFace since it has superior geometric attributes because the angular margin is correspondent to geodesic distance exactly.

$$L_{ArcFace} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{s \cos(\theta_{yi} + m)}}{\sum_{j=1, j \neq yi}^{N} e^{s \cos(\theta_{yi} + m)}}$$  \hspace{1cm} (5)

IV. EXPERIMENTS
A. EXPERIMENTAL CONFIGURATION

In [17] [18], it is observed that the VGG family of networks perform the best among the other deep learning-based architectures in BSL recognition. From the six models, the VGG19 CNN architecture is selected as the feature extractor for carrying out further experiments, including those incorporating angular margin based loss functions. The VGG19 model with SphereFace loss performed well in certain settings and is selected for further analysis. The model is depicted in figure 4.

| Configuration | Train    | Validation | Test    | Category  |
|---------------|----------|------------|---------|-----------|
| Configuration A | Ishara-Lipi Augmented | Ishara-Lipi Augmented | BdSL | Inter-dataset |
| Configuration B | BdSL | BdSL | Ishara-Lipi | Inter-dataset |
| Configuration C | BdSL | BdSL | BdSL | Intra-dataset |
| Configuration D | Ishara-Lipi | Ishara-Lipi | Ishara-Lipi | Intra-dataset |

To evaluate the overall performance of the model configurations, we arrange our evaluation criteria in four configurations shown in table 2. Configuration A and B indicate the experiments which utilize inter-dataset evaluation. For instance, in configuration A, the model is trained and validated on the BdSL dataset before being tested on Ishara-Lipi, while in configuration B, the model is trained and validated on the Ishara-Lipi dataset and tested on the BdSL dataset. Configurations C and D refer to intra-datasets studies, in which training, validation, and testing are all performed on the same datasets.

B. IMPLEMENTATION DETAILS

As previously stated, all of the experiments in this paper are based on the two widely used datasets for BSL. Each dataset is split into three sets train, validation, and test sets with a ratio of 80:10:10. After the train and test set split, the BdSL dataset contains train samples of 19075 samples and 2416 test samples. For augmented Ishara-Lipi data, the training set comprises 18968 samples, and the test set contains 978 images. Initially, we have investigated the works of [17] and [18] by classifying the 35 hand gestures in intra-datasets on the VGG16 network using cross-entropy loss. Afterward, we explore the phenomenon of generalization by assessing inter-dataset performance while maintaining all hyper-parameters constant. For inter-datasets combinations, we have found that setting the batch size to 100 yields better results. All the networks are trained with SGD (Stochastic Gradient Descent) optimizer with a 0.01 learning rate for up to 30 epochs. Training is regularized by fixing weight decay to 5e-4, and batch norm momentum is set as 0.1. All of the experiments are conducted using the Pytorch framework [51]. After analyzing the models, we have modified hyperparameters of three angular margin losses according to our experimental setting. After trial and error, we have observed that angular margin, $m = 3$ achieves the best performance for SphereFace loss [19]. While in both CosFace [41] and ArcFace [41] losses provide better performance in setting margin $m$ as 0.1 with scaling constant $s$ as of value of 2.

| Layers | Train Acc. | Validation Acc. | Test Acc. |
|--------|------------|-----------------|-----------|
| 1000-512-256-128 | 99.36% | 98.42% | 43.62% |
| 512-256-128 | 99.93% | 98.53% | 35.76% |
| 256-128 | 99.96% | 99.79% | 47.79% |

| Layers | Train Acc. | Validation Acc. | Test Acc. |
|--------|------------|-----------------|-----------|
| 1000-512-256-128 | 100% | 99.83% | 49.23% |
| 512-256-128 | 99.99% | 99.79% | 47.34% |
| 256-128 | 100% | 99.75% | 46.52% |

The experiments with hidden layers are mainly focused on configuration A and B, as our primary objective is to improve...
generalization performance; moreover, the accuracy of the remaining two configurations, C and D, are around 99%, as demonstrated in [17] [18]. Initially, we have added four fully connected layers with 1000-512-256-128 neurons on top of the pre-trained network, and then the last fully connected layer, which is 128, is fed to the classification layer. However, we have discovered that increasing the number of neurons makes the model over-fitted which results in poor performance for test accuracy as demonstrated in table 3. As a result, we steadily reduced the number of layers until we discovered that training the model with two layers may significantly improve performance. The drop in accuracy for adding more neurons can be due to the fact that the model memorizes the features particular to the training data and unable to serve them well on new unseen data [52]. It is evident that the fully connected layer with 512 to 256 nodes outperforms other combinations of hidden layers for not only test sets but also validation sets in both configurations, as reported in tables 3 and 4. The findings also indicate that two 256-128 neurons layers do not perform any better than 512-256. The accuracy degrades most likely due to the model’s inability to acquire valuable information from the data for the fewer number of neurons, causing underfitting. Hence, all experiments are carried out using our optimum 512-256 neurons fully connected layers after the pre-trained network.

### B. PERFORMANCE OF CLASSIFICATION

The dataset of [17] and [18] are mainly created for the classification of the BSL. In this subsection, we have analyzed the performance of several architectures for the classification of 35 distinct classes. All six models are evaluated using softmax loss, while all hyper-parameters are held constant for fairness.

1) Model Evaluation on Intra Dataset

Tables 5 and 6 demonstrate the comparative analysis of the six models, along with the proposed approaches of [17] and [18] works.

#### TABLE 5: Performance comparison using several models with configuration C

| Model          | Train Acc. | Validation Acc. | Test Acc. |
|----------------|------------|-----------------|-----------|
| VGG16          | 99.99%     | 99.96%          | 99.63%    |
| VGG16_BN       | 100%       | 99.87%          | 99.57%    |
| VGG19          | 100%       | 99.79%          | 99.59%    |
| VGG19_BN       | 100%       | 99.89%          | 99.57%    |
| ResNet50       | 100%       | 98.40%          | 98.05%    |
| AlexNet        | 99.97%     | 99.37%          | 99.30%    |
| Shafiqul et al. [18] | 99.85% | – | 99.83% |

It can be observed from table 5 that the proposed method of [18] has reported the best results in test accuracy when compared to the other models. Apart from [18] performance, it is clear that VGG16 with 99.63% outperforms the other six models, whereas Resnet50 performs worst than all. It is also evident that VGG16 with batch normalization (VGG16_BN) and VGG19 with batch normalization (VGG19_BN) produce the exact same results in test sets. Performance in table 6 demonstrate that we achieve better performance by augmenting the dataset in comparison to the proposed method of [17]. However, a noticeable observation can be made that the differences of accuracies across all the models are significantly low, as shown in both tables 5 and 6.

2) Model Evaluation on Inter Dataset

We have performed a thorough assessment of the deep learning models trained on inter-datasets (configuration A and B) to determine the best model, as no conclusion can be drawn from the aforementioned findings due to the closer results of the models.

#### TABLE 6: Performance comparison using several models with configuration D

| Model          | Train Acc. | Validation Acc. | Test Acc. |
|----------------|------------|-----------------|-----------|
| VGG16          | 100%       | 99.96%          | 99.63%    |
| VGG16_BN       | 100%       | 99.99%          | 99.63%    |
| VGG19          | 100%       | 99.96%          | 99.63%    |
| VGG19_BN       | 100%       | 100%            | 99.63%    |
| ResNet50       | 100%       | 100%            | 99.63%    |
| AlexNet        | 100%       | 99.83%          | 99.63%    |
| Sanzidul et al. [17] | 92.65% | – | 94.74% |

#### TABLE 7: Accuracy of several models using configuration A

| Model          | Train Acc. | Validation Acc. | Test Acc. |
|----------------|------------|-----------------|-----------|
| VGG16          | 99.99%     | 99.92%          | 48.36%    |
| VGG16_BN       | 100%       | 99.87%          | 45.19%    |
| VGG19          | 99.99%     | 99.87%          | 48.16%    |
| VGG19_BN       | 100%       | 99.92%          | 35.76%    |
| ResNet50       | 100%       | 99.87%          | 30.78%    |
| AlexNet        | 99.99%     | 99.54%          | 34.25%    |

#### TABLE 8: Accuracy of several models using configuration B

| Model          | Train Acc. | Validation Acc. | Test Acc. |
|----------------|------------|-----------------|-----------|
| VGG16          | 100%       | 100%            | 46.11%    |
| VGG16_BN       | 99.99%     | 100%            | 47.60%    |
| VGG19          | 100%       | 100%            | 52.32%    |
| VGG19_BN       | 100%       | 100%            | 35.76%    |
| ResNet50       | 100%       | 99.87%          | 30.76%    |
| AlexNet        | 99.98%     | 99.23%          | 35.74%    |
In our work, we have considered the validation set and training set from the same dataset to evaluate the trained model. Furthermore, we have tested the trained model with different dataset's test set and highlighted the generalization problem. We demonstrate that the validation partition is useful in the intra-dataset as it is drawn from the same dataset as the training set. However, the validation set does not serve its purpose in the inter-dataset, further strengthening our argument that these datasets do not generalize. The results in tables 7 and 8 show that the accuracy of all the networks trained on inter-datasets, which had previously performed well on intra-datasets, has significantly decreased. According to table 7, VGG16 beats all other models for configuration A, while VGG19 is the second-best model with just a 0.2% difference. Table 8 illustrates that VGG19 achieves the best performance so far with the accuracy of 52.32% and outperforms all other models for configuration B. On the other hand, VGG16 performs poorly in configuration B as shown in table 8, albeit achieving maximum accuracy in configuration A. The difference of 0.2% in configuration A between the VGG19 and VGG16 models is negligible compared to the 6.21% difference achieved by VGG19 in configuration B. Therefore, we can conclude that the VGG19 model can generalize well than other models, though there is still scope for improvement. The following experiments are conducted using VGG19, considering it as an optimal model.

Furthermore, we have also observed that VGG16_BN and VGG19_BN perform well enough in intra-datasets. In contrast, their accuracy falls in both configurations A and B, as represented in table 7 and 8. Moreover, ResNet50 has reported the lowest accuracy in configuration A, configuration B, as well as, in configuration C. Hence, ResNet50
architecture is certainly not a suitable model for this analysis, based on all of these studies.

Although the VGG19 model performed better than other CNN models, there is a significant performance gap between the validation and test sets. We demonstrate Grad-CAM visualization to analyze the failure modes of the VGG19 model on inter dataset and identify dataset bias, as shown in figure 5. Grad-CAM technique generates a visual explanation for the transparency of CNN-based models, which localizes class-discriminative regions \[53\]. It is evident from the images in the dataset of original and the inter dataset test images that the images representing the same class differ mainly in angle and orientation. In order to evaluate what wrong decisions the model is making, we conducted inter-dataset and intra-dataset testing, respectively. In both figures 5a and 5b, Grad-CAM visualization of the model revealed that the model has learned to localize the discriminative region for the intra-dataset test, as the model focuses on the center of the important regions. However, the model has misclassified hand gestures for the same class in the inter-dataset test, indicating that the model is unable to concentrate on the crucial regions that help distinguish between classes. These figures provide explicit visual representations of the problem of generalization in biased datasets. The visual representation suggests that the images of these datasets are not diverse and biased in a specific orientation, thus could not highlight the class-discriminative regions to classify correctly.

We have further investigated the reason behind the significant performance gap of intra and inter datasets by plotting the tSNE of VGG19 features. As can be observed from the figure 6, the cluster structure of intra-dataset's feature vectors is much better defined with tighter and more separate clusters, which implies that clusters of the same classes are compacted tightly, and the cluster of distinct hand gestures are separated almost perfectly. However, the learned embedding vectors of the inter-dataset are scattered in the embedding space, implying that the model failed to predict the correct class. Therefore, it is clear that VGG19 extracts more distinct features when using the intra dataset rather than the inter dataset.

3) Generalization Evaluation

The purpose of this study is to signify the importance of generalization while enhancing inter-dataset classification performance. It is mentioned above that inter-dataset evaluation is to train a model in one dataset while testing on another
dataset to show the efficacy in the generalization of the deep learning models. But, from the above findings, it is observed that training on inter-dataset reduces the performance of all the networks.

**TABLE 9: Performance of the different loss functions using configuration A**

| Loss Function | Train Acc. | Validation Acc. | Test Acc. |
|---------------|------------|-----------------|-----------|
| ArcFace       | 99.95%     | 99.49%          | 51.43%    |
| CosFace       | 99.94%     | 99.24%          | 51.02%    |
| SphereFace    | 99.98%     | 99.79%          | **55.93%**|
| SoftMax       | 99.99%     | 99.87%          | 48.16%    |

Therefore, this subsection only focuses on inter-dataset classification rather than intra-dataset classification because the accuracy of intra-datasets classification is consistently around 99%, leaving us with little opportunity for improvement. A comparative analysis of loss functions trained on VGG19 model are illustrated in tables 9 and 10. As shown in table 9, the performance of VGG19 is enhanced by a considerable amount after utilizing angular loss functions using the dataset of Configuration A. For this configuration, SphereFace loss achieves the best performance so far, reporting accuracy of 55.93% in the test set. In addition, the accuracy difference between SphereFace and softmax is 7.77%, which is a remarkable improvement. For configuration B, the performance of SphereFace reveals that it has about 48% accuracy, which is less than the accuracy of softmax of 52.32%, as shown in table 10. We can state a few observations based on performance analysis of loss functions:

- **It is worth mentioning that configuration B has a comparatively smaller test set than configuration A.** As we

**FIGURE 8: Confusion Matrix of model using Configuration B**

**TABLE 10: Performance of different loss functions using configuration B**

| Loss Function | Train Acc. | Validation Acc. | Test Acc. |
|---------------|------------|-----------------|-----------|
| ArcFace       | 100%       | 100%            | 44.58%    |
| CosFace       | 100%       | 99.96%          | 44.91%    |
| SphereFace    | 100%       | 100%            | 47.81%    |
| SoftMax       | 100%       | 100%            | **52.32%**|
have seen, SphereFace performs well on a large test set with a performance improvement of almost 8%. In configuration B, SphereFace performs poorly than SoftMax. However, considering the large test set in configuration A, it might be possible that the model is overfitted in the small test set as configuration B.

- For configuration A, ArcFace and CosFace have almost the same performance, which is still better than softmax, indicating that softmax shows the worst performance for this specific setting.
- Arcface and CosFace both performed poorly in configuration B compared to other loss functions.

Therefore, we can conclude that SphereFace loss infers to be the best loss function for inter-dataset classification. Additionally, we can state that VGG19 with the SphereFace loss is the optimal solution for our generalization model.

TABLE 11: Performance of individual class using configuration A and B

| Class | Alphabet | Acc. of Config. A | Acc. of Config. B |
|-------|----------|------------------|------------------|
| 1     | a        | 8.5%             | 80.56%           |
| 2     | å        | 11.3%            | 37.04%           |
| 3     | i        | 69.0%            | 15.38%           |
| 4     | u        | 22.9%            | 28.12%           |
| 5     | ê        | 0.0%             | 64.29%           |
| 6     | ö        | 52.9%            | 37.93%           |
| 7     | ka       | 34.8%            | 18.32%           |
| 8     | kha      | 73.3%            | 61.29%           |
| 9     | ga       | 78.9%            | 71.43%           |
| 10    | gha      | 46.5%            | 30.43%           |
| 11    | ca       | 95.7%            | 82.76%           |
| 12    | cha      | 51.4%            | 22.58%           |
| 13    | ja       | 6.9%             | 85.19%           |
| 14    | jha      | 31.8%            | 83.33%           |
| 15    | ta       | 62.9%            | 59.26%           |
| 16    | t ha     | 82.5%            | 64.52%           |
| 17    | d a      | 71.2%            | 29.17%           |
| 18    | dha      | 53.0%            | 17.24%           |
| 19    | ta       | 44.2%            | 29.63%           |
| 20    | tha      | 22.9%            | 89.66%           |
| 21    | da       | 33.8%            | 76.67%           |
| 22    | dha      | 32.9%            | 56.00%           |
| 23    | na       | 68.7%            | 58.62%           |
| 24    | pa       | 73.9%            | 24.00%           |
| 25    | pha      | 44.1%            | 77.78%           |
| 26    | ba       | 7.4%             | 78.57%           |
| 27    | bha      | 70.4%            | 75.86%           |
| 28    | ma       | 52.4%            | 55.17%           |
| 29    | ya       | 98.6%            | 88.00%           |
| 30    | ra       | 27.5%            | 14.81%           |
| 31    | sa       | 66.7%            | 16.67%           |
| 32    | ha       | 40.3%            | 84.62%           |
| 33    | f        | 73.2%            | 81.48%           |
| 34    | ñ        | 65.22%           | 50.00%           |
| 35    | h        | 64.29%           | 61.54%           |

Figures 7 and 8 illustrates the confusion matrix of the classification of hand gestures using modified VGG19 network with SphereFace loss of configuration A and B, respectively, for a more detailed interpretation of generalization. The confusion matrix is able to describe the performance for each class in classification, where the scores along the diagonal represent the correctly classified predicted labels to the output class. From the recognition of alphabets for configuration A, it can be deduced that model is unable to recognize the appropriate class of class 5 i.e. "ê", as demonstrated in figure 7. In figure 8 the confusion matrix of configuration B is demonstrated, where "pa" and "i" classes yield the worst results. In this illustration, class "ê" produces better results than configuration A, whereas class "i" shows poor performance. We can deduce from these findings that the sign of "ê" and "i" are quite identical, so the model could not distinguish both at the same time. We have also noticed that some classes are misclassified as well. Along with the confusion matrix, we have presented the accuracy of each class in table 11 for a clear representation of both configurations. According to table 11, "ya" and "tha" have the best accuracy scores of all the classes, with 98.6% and 89.66% in configuration A and B, respectively. However, the prediction accuracy of these classes is comparatively poor. It can also be observed that 19 alphabets for configuration A and 22 alphabets of configuration B have an accuracy greater than 50%. In conclusion, the analysis of our research results provides evidence that, while intra-dataset results are encouraging, there is a significant performance gap in inter-datasets.

VI. DISCUSSION

A. COMPARISON OF DATASETS

After pre-processing the data, it is observed that the BdSL dataset is significantly deficient in diversity since many of the data are almost identical. Although the data was obtained from 25 people, they augmented it by rotating, zooming, and shifting, which is likely why the images are similar. In
accuracy. As we are concerned with the data, we processed the data by converting the RGB image to GrayScale. However, it performed considerably worse in the VGG16 model, with an accuracy of 38%. We have also implemented a Laplacian filter, i.e., an edge detector, to compute the second derivatives of an image. In addition, we also sharpened the images for overcoming blurring to enhance the pixel value of the edge pixels, whose gray value tends to be higher. However, the performance of the models suffered after these pre-processings. Therefore, we only used normalized images in the models. In the VGG16 model, we have used dropout layers with $p$ value set to 0.25, which were later removed as they lowered accuracy. We have also attempted freezing the convolution layers of the VGG models, but the model could not learn anything.

Furthermore, figure [11] demonstrates the two-dimensional representation of feature distributions in four different configurations. We can observe that the cluster structures of intra-dataset’s feature vectors are much well defined due to angular margin, where intra-class compactness and inter-class diversity are evident. However, numerous misclassifications can be observed in inter-dataset cluster structures resulting in ambiguity in decision boundaries, while all of the different classes’ feature vectors are compact together.

| Configuration | Cosine Similarity | Euclidean Distance |
|---------------|-------------------|--------------------|
| A             | 11.4% (4/35)      | 11.4% (4/35)       |
| B             | 14.3% (5/35)      | 11.4% (4/35)       |
| C             | 23% (8/35)        | 23% (8/35)         |
| D             | 20% (7/35)        | 20% (7/35)         |

We have also computed Cosine similarity and Euclidean distance metrics using the feature vectors. The accuracy score of Cosine similarity and Euclidean distance of intra and inter datasets are presented in table [12]. Both techniques have shown unsatisfactory performance. The intra and inter datasets have an accuracy of range in 20-23% percent and 11-14% percent, respectively.

VIII. CONCLUSION

Recognition of Sign Language has been a popular research field for the Deaf and Mute community. Although Bangla is a primary language with a large deaf and mute population, there has been limited research on Bangla Sign Language (BSL) recognition. This paper demonstrates the importance of generalization in finger-spelled BSL recognition by utilizing several deep learning models and angular loss func-
tions to improve inter-dataset performance. Among all the experiments, we have found that VGG-19 architecture with SphereFace loss function has shown optimal performance achieving 55.93% and 47.81% on inter-dataset. However, the experimental results are not promising for inter-dataset configurations that have been reported in the intra-dataset findings. Our research has discovered that while the BSL dataset may be large enough for recognition tasks, it lacks diversity, which is vital for generalization. As well as, the deep learning model tends to be underfitting, especially with a short dataset like Ishara-Lipi. To summarize, it is vital to develop a large and diversified dataset for BSL that is more generic and can be deployed in real-world systems and serve the deaf and mute population.

REFERENCES
[1] M. Alauddin and A. H. Joarder, “Deafness in bangladesh,” in Hearing Impairment. Springer, 2004, pp. 64–69.
[2] R. G. Rajan and M. J. Leo, “American sign language alphabets recognition using hand crafted and deep learning features,” in 2020 International Conference on Inventive Computation Technologies (ICICT). IEEE, 2020, pp. 430–434.
[3] R. Bowden12, A. Zisserman, T. Kadir, and M. Brady, “Vision based interpretation of natural sign languages,” 2003.
[4] B. Shi, A. M. Del Rio, J. Keane, J. Michaux, D. Brentari, G. Shakhnarovich, and K. Livescu, “American sign language fingerspelling recognition in the wild,” in 2018 IEEE Spoken Language Technology Workshop (SLT). IEEE, 2018, pp. 145–152.
[5] B. Kang, S. Tripathi, and T. Q. Nguyen, “Real-time sign language fingerspelling recognition using convolutional neural networks from depth map,” in 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR). IEEE, 2015, pp. 136–140.
[6] N. Pageault and R. Bowden, “Spelling it out: Real-time asl fingerspelling recognition,” in 2011 IEEE International conference on computer vision workshops (ICCV workshops). IEEE, 2011, pp. 1114–1119.
[7] L. Rioux-Maladague and P. Giguere, “Sign language fingerspelling classification from depth and color images using a deep belief network,” in 2014 Canadian Conference on Computer and Robot Vision. IEEE, 2014, pp. 92–97.
[8] S. Ameen and S. Vadera, “A convolutional neural network to classify american sign language fingerspelling from depth and color images,” Expert Systems, vol. 34, no. 3, p. e12197, 2017.
[9] D. Aryanie and Y. Heryadi, “American sign language-based fingerspelling recognition,” in 2020 International Conference on Inventive Computation Technologies (ICICT). IEEE, 2020, pp. 430–434.
recognition using k-nearest neighbors classifier,” in 2015 3rd International Conference on Information and Communication Technology (CoICT), IEEE, 2015, pp. 533–536.

[10] Y. Hu, H.-F. Zhou, and Z.-G. Wang, “Sign language fingerpost recognition using depth information and deep belief networks,” International Journal of Pattern Recognition and Artificial Intelligence, vol. 32, no. 06, p. 1850018, 2018.

[11] P. Nakjai and T. Katanyukul, “Hand sign recognition for Thai finger spelling: An application of convolution neural network,” Journal of Signal Processing Systems, vol. 91, no. 2, pp. 131–146, 2019.

[12] T. Parwat and P. Serasingalakal, “Thai finger-spelling sign language recognition using global and local features with svm,” in 2017 9th international conference on knowledge and smart technology (KST). IEEE, 2017, pp. 116–120.

[13] X. Jiang, M. Lu, and S.-H. Wang, “An eight-layer convolutional neural network with stochastic pooling, batch normalization and dropout for fingerpost recognition of chinese sign language,” Multimedia Tools and Applications, vol. 79, no. 21, pp. 15607–15615, 2020.

[14] S. Aly, B. Osman, W. Aly, and M. Saber, “Arabic sign language fingerpost recognition from depth and intensity images,” in 2016 12th International Computer Engineering Conference (ICENCE). IEEE, 2016, pp. 99–104.

[15] D. Dahmani and S. Larabi, “User-independent system for sign language finger spelling recognition,” Journal of Visual Communication and Image Representation, vol. 25, no. 5, pp. 1240–1250, 2014.

[16] I. Chowdhury, “Randomnet: The 100 most spoken languages around the world,” Feb 2020. [Online]. Available: https://www.visualcapitalist.com/100-most-spoken-languages/

[17] M. S. Islam, S. S. S. Mousumi, N. A. Jessan, A. S. A. Rabby, and S. A. Hossain, “Ishara-bochon: The first multipurpose open access dataset for bangla sign language isolated digits,” in International Conference on Recent Trends in Image Processing and Pattern Recognition. Springer, 2018, pp. 420–428.

[18] M. S. Islam, S. S. S. Mousumi, A. S. A. Rabby, and S. A. Hossain, “A simple and mighty arrowhead detection technique of bangla sign language characters with cnn,” in International Conference on Recent Trends in Image Processing and Pattern Recognition. Springer, 2018, pp. 429–437.

[19] M. S. Islam, S. S. S. Mousumi, N. A. Jessan, A. S. A. Rabby, and S. A. Hossain, “Ishara-lipi: The first complete multipurposeopen access dataset of isolated characters for bangla sign language,” in 2018 International Conference on Bangla Speech and Language Processing (ICBSLP). IEEE, 2018, pp. 1–4.

[20] S. Islam, S. S. S. Mousumi, A. S. A. Rabby, S. A. Hossain, and S. Abujar, “A potent model to recognize bangla sign language digits using convolutional neural network,” Procedia computer science, vol. 143, pp. 611–618, 2018.

[21] T. Hossain, F. S. Shishir, and F. M. Shah, “A novel approach to classify bangla sign digits using capsule network,” in 2019 22nd International Conference on Computer and Information Technology (ICCIT). IEEE, 2019, pp. 1–6.

[22] M. M. Hasan, A. Y. Srizon, and M. A. M. Hasan, “Classification of bengali sign language characters by applying a novel deep convolutional neural network,” in 2020 IEEE Region 10 Symposium (TENSYMP). IEEE, 2020, pp. 1303–1306.

[23] R. A. Nihal, S. Rahman, N. M. Broi, and S. A. Deowan, “Bangla sign alphabet recognition with zero-shot and transfer learning,” Pattern Recognition Letters, 2021.

[24] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, “Sphereface: Deep hypersphere embedding for face recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 212–220.

[25] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, “Cosface: Large margin cosine loss for deep face recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 5265–5274.

[26] P. Verma and D. C. Verma, Bengali (Bangla) alphabet book. Chanda Books, 2008.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., “Imagenet large scale visual recognition challenge. cor,” arXiv preprint arXiv:1409.0575, vol. 372, 2014.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Advances in neural information processing systems, vol. 25, pp. 1097–1105, 2012.

[29] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[30] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.

[31] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapasakis, “Deep learning for computer vision: A brief review,” Computational intelligence and Neuroscience, vol. 2018.

[32] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in International conference on machine learning. PMLR, 2015, pp. 448–456.
A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-

[50] F. Wang, J. Cheng, W. Liu, and H. Liu, “Additive margin softmax for face verification,” IEEE Signal Processing Letters, vol. 25, no. 7, pp. 926–930, 2018.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-

[52] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, http://www.deeplearningbook.org.

[53] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 618–626.

[54] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[55] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” arXiv preprint arXiv:1704.04861, 2017.

[56] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, “Mnasnet: Platform-aware neural architecture search for mobile,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2820–2828.

[57] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
NABEEL MOHAMMED received a bachelor’s degree in computer science from Monash University, Australia. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, North South University. He worked as a Software Developer for Editure Ltd., a Melbourne based Software Firm specializing in providing software solutions for K-12 schools. After three and a half years in that role he moved back into academia to complete the Ph.D. degree at Monash University, Australia.