A Novel Approach to Classify Bangla Sign Digits using Capsule Network

Tonmoy Hossain, Fairuz Shadmani Shishir, Faisal Muhammad Shah
HELLO!

I AM TONMOY HOSSAIN

Lecturer
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology
INTRODUCTION
INTRODUCTION

- Hearing impaired refers to as partial or complete inability to hear
- Approximately 13 million people are suffering variable degrees of hearing loss[1]
- Previously, traditional machine learning technique was used
- Capsule Network is introduced for the classification task

[1]. Amin MN: Prevention of Deafness and Primary Ear Care (Bengali)- Society for Assistance to Hearing Impaired Children (SAHIC), Mohakhali, Dhaka-1212, Bangladesh.
MOTIVATION

- Well adaptation of automated sign digits classification in the perspective of Bangladesh
- Developing practical application for the deaf people
- On the perspective to the people who are unable to speak
RESEARCH
DOMAIN
Problem

Classification of Bangla Sign Digits

How we can implement the problem?

- Traditional Machine Learning Techniques
- Image Processing Methods
- Deep Learning Model
BACKGROUNDs
Sign Language

- Use visual-manual modality to convey meaning
- Expressed through manual articulations in combination with non-manual elements
- Generally more than 137 types of sign language used throughout the world
Bangla Sign Digits

Fig 1: A depiction of Bangla Sign Digits
Capsule

- A group of neurons (vectors) specifying the feature of the object and its likelihood
- Activity vector represents the instantiation parameter of the entity
  1. Length of the activity vector represents the existence of the entity
  2. Orientation to represent the instantiation parameters
BACKGROUND STUDIES
Existing Works

- **Sanzidul et al. 2018**
  - A convoluted 22 layer ConvNet architecture was implemented and achieved 94.88% of accuracy

- **Bikash et al. 2012**
  - An ensemble method of negative correlation learning and feature extraction was employed and 93% of accuracy was accomplished

- **Shahjalal et al. 2019**
  - Tracking, detecting and recognizing are the primary steps of the model which is based one data augmentation

- **Sinith et al. 2012**
  - Support Vector Machine along with Binary tree concept was operated for the classification
Proposed Methodology
Fig 2: Proposed methodology for classification using Capsule Network
Dataset

- THE ISHARA-LIPI DATASET
- Total Images: 1000
- Break down into ten categories based on the digits
- All the images are gray scaled and binary colored
- An identical shape of 128 x 128 pixels is maintained in all the images

Some Examples
Dataset Preparation and Pre-processing

- A total of 1000 images partitioning into 10 classes each of 100 images
- All images are converted into 28×28 pixels
- Images are labeled after binarization
- Converted the image pixels into a CSV file
Input Layer

- Pre-processed images are working as the input
- Dimension: 28×28×1
- Output a unit vector of size 10
Convolutional Layer - 1

- Input Size: 28x28x1
- Output Size: 20x20x256
- Filter or Kernel Size: 9 and No padding is done
- Rectified Linear Unit (RELU) is used as the activation function
- Preserve Spatial Relation between the pixels

Output Size = $\left(\frac{n + 2 \times p - f}{2} + 1\right) \times \left(\frac{n + 2 \times p - f}{2} + 1\right)$
Convolutional Layer - 2

- Input Size: $20 \times 20 \times 256$
- Output Size: $6 \times 6 \times 256$
- Obtained a feature map after the convolutional layer
Reshape and Lambda Layer

- Primary capsule layer
- Constituted with the feature map of capsules and
- Affine transformation, weighted sum is operated
- Activation Function: Squashing Function (Non linear)

Reshape Layer
6 × 6 × 256

Lambda Layer
6 × 6 × 8 × 32
Properties of Primary Capsule Layer

- Matrix Multiplication of input vectors with weight matrices
- Weighting input vectors
- Weighted sum
- Squashing Function
Capsule Layer (digitcaps layer)

- The higher level capsule layer
- Generate the final feature map
Length Layer and Classification

- Generate the final feature map
- Return the exact input shape as a tensor
## A brief workflow of the proposed model

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pre-processing and converted into CSV</td>
</tr>
<tr>
<td>2</td>
<td>Transform into an input vector</td>
</tr>
<tr>
<td>3</td>
<td>A feature map is obtained in the convolutional layers</td>
</tr>
<tr>
<td>4</td>
<td>Affine transformation is applied</td>
</tr>
<tr>
<td>5</td>
<td>Apply weighted sum</td>
</tr>
<tr>
<td>6</td>
<td>Activation function — Squashing is operated</td>
</tr>
<tr>
<td>7</td>
<td>A vector (shrunked) is sent to the capsule layer</td>
</tr>
<tr>
<td>8</td>
<td>Dynamic Routing Algorithm is performed in the capsule layer</td>
</tr>
<tr>
<td>9</td>
<td>Loss function is calculated for each capsule and sum up for the final loss</td>
</tr>
<tr>
<td>10</td>
<td>Final classified output vector is assembled</td>
</tr>
</tbody>
</table>
Experimental Results
Experimental Setup

☑ Train the data into three types of splitting ratio: 70:30, 80:20 and 90:10
☑ Best Result: 90:10 (80% training, 10% validation and 10% testing)
☑ Google Colab is used to train the model
## Dimension of the Network Architecture

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>Input Shape</th>
<th>Output Shape</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>(28, 28, 1)</td>
<td>(28, 28, 1)</td>
<td>0</td>
</tr>
<tr>
<td>Convolutional Layer – 1</td>
<td>(28, 28, 1)</td>
<td>(20, 20, 256)</td>
<td>17712</td>
</tr>
<tr>
<td>Convolutional Layer – 2</td>
<td>(20, 20, 256)</td>
<td>(6, 6, 256)</td>
<td>4479232</td>
</tr>
<tr>
<td>Reshape</td>
<td>(6, 6, 256)</td>
<td>(1152,8)</td>
<td>0</td>
</tr>
<tr>
<td>Lambda</td>
<td>(1152,8)</td>
<td>(1152,8)</td>
<td>0</td>
</tr>
<tr>
<td>Capsule Layer</td>
<td>(1152,8)</td>
<td>(10, 16)</td>
<td>1486080</td>
</tr>
<tr>
<td>Length Layer</td>
<td>(10, 16)</td>
<td>(10)</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Dimension of all layers of the network architecture
## Hyperparameter Setup

<table>
<thead>
<tr>
<th>No</th>
<th>Hyper-parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initialization_kernel</td>
<td>glorot_uniform</td>
</tr>
<tr>
<td>2</td>
<td>Initialization_bias</td>
<td>zeros</td>
</tr>
<tr>
<td>3</td>
<td>Learning_rate</td>
<td>0.001</td>
</tr>
<tr>
<td>4</td>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>5</td>
<td>Batch Sie</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Epoch</td>
<td>50</td>
</tr>
<tr>
<td>7</td>
<td>Steps per epoch</td>
<td>580</td>
</tr>
</tbody>
</table>

Table 2: Hyperparameters of the architecture
Classification

Table 3: Classification of the proposed model

<table>
<thead>
<tr>
<th>No</th>
<th>Splitting Ratio</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>70:30</td>
<td>93.92%</td>
</tr>
<tr>
<td>2</td>
<td>80:20</td>
<td>98.25%</td>
</tr>
<tr>
<td>3</td>
<td>90:10</td>
<td>98.84%</td>
</tr>
</tbody>
</table>

Table 3: Classification of the proposed model
## Performance Comparison

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Layer</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sanzidul et al.</td>
<td>22</td>
<td>95.5%</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>7</td>
<td>98.84%</td>
</tr>
</tbody>
</table>

Table 4: Performance Comparison with the existing work on same dataset
CONCLUSION
Conclusion

- The model is a seven-layer architecture which is an unsophisticated model to train and test.
- Computation power and training time is curtailed with respect to the existing articles as well as data augmentation is done.
- Spatial properties of an object are taken into account along with the activity vector.
- Pooling layer which is lossy and does not conserve all the spatial information is apprehended in this architecture.
FUTURE PLAN
Future Plan

- Work on Bangla Sign Character
- Expand the model to work on Videos
- Employ the model on a substantial large dataset
- Try to generate text from sign language conversation video
THANK YOU!