American Sign Language Assistant

Presented To:

• Dr. Christopher Pollett
• Dr. Robert Chun
• Shruti Kothari

Presented By:

Charulata Lodha | Spring 2021
Department of Computer Science
San Jose State University
Agenda

- Introduction
- Background
- Implementation
- Related Work
- Design
- Datasets
- Experiments and Results
- Conclusions
- References
Introduction

• Between 6 and 8 million people in the United States have some form of language impairment [1].

• Every situation can be stressing for a person with disabilities in a confined and busy place.

• The Americans with Disabilities Act (ADA) is a civil rights law that prohibits any kind of discrimination based on disability.

• The objective of this project is to build a digital assistance system to help deaf community in shopping center.
The digital assistance system provides equivalent shopping experience to deaf community as the one with the ability to speak English

- Enable live communication between shop keeper and deaf customers
- Develop a base for building contactless and ASL signs based kiosk at shopping centers
- Leverage Convolution Neural Networks and Unity Game Engine

Background

• American Sign Language (ASL) is one of the leading sign language used in the U.S. [2].

• Existing accessibility technologies:
  • HoloHear: user speak ASL aloud to a 3D holographic model
  • TapSOS: connect with emergency services in a nonverbal way
  • Amplifier (FM systems): to fully understand the advice of the sales assistant
Background: Shopping Center ASL

- ASL has hand gestures involves static as well as motion gestures.
- 3-Dimensional analysis is done with the perspective of building an AI model

<table>
<thead>
<tr>
<th>Sign Category</th>
<th>Signs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 D Motion ASL Gesture</td>
<td>Red, Yellow, Hand Wave, Okay,</td>
</tr>
<tr>
<td>3 D Motion ASL Gesture</td>
<td>Blue, Pay, Hello, Eat, Thank You, Offer</td>
</tr>
</tbody>
</table>

Figure 1: Red Sign in ASL
Figure 2: Pay Sign in ASL
Figure 3. ASL Alphabets

Implementation: LeNet – 5 CNN Model

• Our project aims at finding visual patterns in temporal representation of ASL action sequences.

• Yann LeCun showed that minimizing the number of free parameters in neural networks can enhance the generalization ability of neural networks.

• It is widely used for various image recognition problems.
Implementation: LeNet – 5 CNN Architecture

```python
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3), padding='same', activation='relu', input_shape=(64, 64, 3)))
model.add(MaxPool2D(strides=2))
model.add(Conv2D(filters=64, kernel_size=(3, 3), padding='valid', activation='relu'))
model.add(MaxPool2D(strides=2))
model.add(Flatten())
model.add(Dense(256, activation='sigmoid'))
model.add(Dense(84, activation='relu'))
model.add(Dense(9, activation='softmax'))
```
Implementation: LeNet – 5 CNN Model Summary

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv2d (Conv2D)</td>
<td>(None, 64, 64, 32)</td>
<td>896</td>
</tr>
<tr>
<td>max_pooling2d (MaxPooling2D)</td>
<td>(None, 32, 32, 32)</td>
<td>0</td>
</tr>
<tr>
<td>conv2d_1 (Conv2D)</td>
<td>(None, 30, 30, 64)</td>
<td>18496</td>
</tr>
<tr>
<td>max_pooling2d_1 (MaxPooling2)</td>
<td>(None, 15, 15, 64)</td>
<td>0</td>
</tr>
<tr>
<td>flatten (Flatten)</td>
<td>(None, 14400)</td>
<td>0</td>
</tr>
<tr>
<td>dense (Dense)</td>
<td>(None, 256)</td>
<td>3686656</td>
</tr>
<tr>
<td>dense_1 (Dense)</td>
<td>(None, 84)</td>
<td>21588</td>
</tr>
<tr>
<td>dense_2 (Dense)</td>
<td>(None, 9)</td>
<td>765</td>
</tr>
</tbody>
</table>

Total params: 3,728,401
Trainable params: 3,728,401
Non-trainable params: 0
Implementation: Support Vector Machine
Machine Learning Model

• SVM is supervised machine learning algorithm
• It is widely used for classification and regression analysis.
• SVM can produce results on complex datasets that are smaller in size.
• Our datasets are temporal images that are very complex for general object recognition model.

SVM Hyperparameter

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>penalty parameter to control error rate</td>
</tr>
<tr>
<td>Gamma</td>
<td>the kernel coefficient</td>
</tr>
<tr>
<td>Kernel</td>
<td>linear, sigmoid, rbf, and poly</td>
</tr>
</tbody>
</table>

Figure 5: SVM Parameters for our model

```python
param_grid = [
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
    {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
]
svc = svm.SVC()
clf = GridSearchCV(svc, param_grid)
clf.fit(X_train, y_train)
```

Figure 5: SVM Parameters for our model
Implementation : Unity Game Engine

We leveraged animations on humanoid avatar in Unity Game Engine for gesture dataset generation.

Typical Steps Involved :
1. Designing shop with items and humanoid avatars in 3D Unity Scene
2. Rescaling objects
3. Configuring Humanoid Rigs: Skeleton and Muscle Setting
4. Adding Animation Properties to Bone Points
5. Build custom gesture animation using Dopesheets and Curves
6. Designing Workflow/Animation State Diagram of events
8. IK Scriptwriting for object movement & camera placement
9. Light Settings in Unity
10. Mp4 video generation using Unity Recorder

Figure 6: ASL Gesture by Humanoid Avatar in Unity
Implementation: Unity Game Engine

Figure 7: Bone Motion Property in Animation Configuration

Figure 8: Okay Sign Curves Configuration

Figure 9: Okay Sign Dopesheet Configuration
Implementation: Unity Game Engine

Figure 10: Unity Scene: Customer doing Okay ASL Gesture Sign in a shop setting
Implementation : OpenPose

- OpenPose is a real time approach for multi-person key point detection: body, foot, hand, and facial key points.

- For this project, we used 2D real-time multi-person key point detection, the output of which is being fed to get the 3D temporal mapping of skeleton frames.

- It outputs 25 body part locations (x, y) and detection confidence (c) formatted as x0,y0,c0,x1,y1,c1....
Implementation : OpenPose
Implementation: Temporal Representation of 3D Skeletal Action Sequences

- The 3D skeleton is mapped into RGB color palettes.
- Its body invariant approach.
- We followed the approach proposed in [3] Sohaib, et al for transforming skeleton sequences into a temporal RGB image.
- This approach reduces the high dimensionality of action sequences to just image classification.

Figure 13: Skeleton to RGB Transformation

Figure 14: 3D coordinates to RGB Mapping
Implementation: Temporal Representation of 3D Skeletal Action Sequences

- High Level Steps:
  - Input: json/.skeleton skeleton files
  - Parse the input file to get the co-ordinates into numpy X,Y,Z matrix where each matrix has the shape = Number of Joint * Number of Frames
  - Normalized the X,Y,Z Matrix using Head Body Joint
  - RGB Channel Generation
  - Normalization of this to map to a range of 0 – 255 color value
  - Temporal Image Generation of size = 64*64

```python
for i in range(self.data.shape[0]):
    self.data[i,:, 0] = self.data[self.norm, :, 0]
    self.data[i, :, 1] = self.data[self.norm, :, 1]
    self.data[i, :, 2] = self.data[self.norm, :, 2]
```

Figure 15: Normalization for 3D co-ordinate

```python
self.data[:, :, 0] = (self.data[:, :, 0] - min_x) / (max_x - min_x)
self.data[:, :, 1] = (self.data[:, :, 1] - min_y) / (max_y - min_y)
self.data[:, :, 2] = (self.data[:, :, 2] - min_z) / (max_z - min_z)
```

Figure 16: Normalization for 2D co-ordinate
Implementation: Temporal Representation Examples

Figure 17: Human Jump (Sample output temporal image)

Figure 18: Human Hop (Sample output temporal image)

Figure 19: Human Hand Wave (Sample output temporal image)

Figure 20: Skeleton Format

\[
p_{\text{norm}} = \text{floor} \left( 255 \times \frac{p - c_{\text{min}}}{c_{\text{max}} - c_{\text{min}}} \right)
\]
Related Work

• “British Sign Language Recognition via Late Fusion of Computer Vision and Leap Motion with Transfer Learning to American Sign Language” by Bird, J.J.; Ekárt, A.; Faria, D.R
  • fusion approach to multi-modality in sign language recognition

• “Skeleton-based Action Recognition with Convolutional Neural Networks” by Chao Li, Qiaoyong Zhong, Di Xie, Shiliang Pu
  • first to adapt Faster R-CNN to the task of skeleton-based temporal action detection

• “3D Skeleton-Based Action Recognition by Representing Motion Capture Sequences as 2D-RGB Images” by Sohaib Laraba, Med Brahimi, Joëlle Tilmanne and Thierry Dutoit
  • present a new representation of motion sequences Seq2Im for sequence to image which projects motion sequences onto the RGB domain.
Design: Application Design

Figure 21: Application Design
Design : Project Architecture

Figure 22 : Application Design
Datasets Overview

• MNIST ASL Alphabets Dataset
• Unity Okay Animation Dataset
• NTU RGB Action Recognition Dataset
• ASL Leap Motion Controller Dataset
Datasets: MNIST ASL Alphabets Recognition

- It matches closely with the classic MNIST.
- This dataset is provided as a .csv file that has 1 column for label and 784 columns for pixel values of the gesture image.
- Class label from 0 to 25 map to A-Z.
- Motion gestures J=9 or Z=25 excluded.
- Training data size = 27,455 and Test data size = 7,172

Figure 23: MNIST ASL Dataset Preview
Datasets: Unity Okay Animation Dataset

- This is the synthetic video dataset generated by us using Unity Animation where a person is performing an ‘Okay’ sign in ASL.

- The dataset has 50 videos taken from varied angles in the 3D Unity Scene.

- Variety is added through different background contrast, various angles of camera, light settings and through addition of shop objects.
Datasets : NTU RGB+D Action Recognition Dataset

- It contains videos captured from Kinect V2 cameras providing RGB videos, 3D skeletal data, depth map sequences for each video sample.
- Class labels from 1 - 60 to represent respective action classes.
- Total dataset = 56,880 videos and .skeleton files

Figure 24: NTU RGB Action Classes
Datasets: NTU RGB Action Recognition Dataset

Figure 25: NTU RGB Skeleton for an Action Class
Datasets : ASL Leap Motion Controller Dataset

- This dataset contains 25 subjects performing 60 different signs of the ASL and includes more than 17,000 signs in total.

- The dataset contains ASL gesture classes like red, blue, yellow, come, cost, shop, big, small and others that are widely used in a shop setting and hence is idle for our project.

Figure 26: Bone Point Representation
Datasets: ASL Leap Motion Controller Dataset

• Every action has a separated dataset file for left and right hand.

• Each of these files represent the skeleton motion trajectory in 3D for that single hand.

<table>
<thead>
<tr>
<th>Time</th>
<th>thumbProximal_L_X</th>
<th>thumbProximal_L_Y</th>
<th>thumbProximal_L_Z</th>
<th>thumbDistal_L_X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.1196449999999998</td>
<td>0.02544089</td>
<td>0.4060230999999997</td>
<td>0.1184248000000001</td>
</tr>
<tr>
<td>1</td>
<td>0.1194022</td>
<td>0.02560966</td>
<td>0.4058299999999997</td>
<td>0.1180625</td>
</tr>
<tr>
<td>2</td>
<td>0.1191847</td>
<td>0.02572824</td>
<td>0.4053169000000004</td>
<td>0.1177519</td>
</tr>
<tr>
<td>3</td>
<td>0.1188636</td>
<td>0.02577735</td>
<td>0.4050539999999997</td>
<td>0.11715899999999999</td>
</tr>
<tr>
<td>4</td>
<td>0.1180744</td>
<td>0.02620117</td>
<td>0.4046636</td>
<td>0.116481</td>
</tr>
</tbody>
</table>

Figure 27: LMC Orange Left Hand Dataset Preview

<table>
<thead>
<tr>
<th>Time</th>
<th>thumbProximal_L_X</th>
<th>thumbProximal_L_Y</th>
<th>thumbProximal_L_Z</th>
<th>thumbDistal_L_X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.1097075</td>
<td>0.01622243</td>
<td>0.3975657999999997</td>
<td>-0.1132996000000001</td>
</tr>
<tr>
<td>1</td>
<td>-0.1098230000000002</td>
<td>0.01601733</td>
<td>0.3980123</td>
<td>-0.1135501</td>
</tr>
<tr>
<td>2</td>
<td>-0.1100047</td>
<td>0.01598163</td>
<td>0.3962055999999994</td>
<td>-0.11370119999999999</td>
</tr>
<tr>
<td>3</td>
<td>-0.1098692</td>
<td>0.01586817</td>
<td>0.3983150000000003</td>
<td>-0.1135689999999998</td>
</tr>
<tr>
<td>4</td>
<td>-0.1098382000000001</td>
<td>0.01578118</td>
<td>0.3984414</td>
<td>-0.11358199999999998</td>
</tr>
<tr>
<td>5</td>
<td>-0.1098796000000001</td>
<td>0.01576672</td>
<td>0.3968385000000003</td>
<td>-0.1136150000000001</td>
</tr>
</tbody>
</table>

Figure 28: LMC Orange Right Hand Dataset Preview
Experiments and Results

• ASL Alphabet Recognition

• NTU-RGB+D on SVM & LeNet-5 model

• Leap Motion Controller on SVM & LeNet-5 model

• Impact on accuracy 2d vs 3d co-ordinates
Experiments: ASL Alphabets Recognition

Figure 29: ASL Live Alphabet Recognition Preview

```
Model: "sequential"

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv2d (Conv2D)</td>
<td>(None, 28, 28, 6)</td>
<td>156</td>
</tr>
<tr>
<td>max_pooling2d (MaxPooling2D)</td>
<td>(None, 14, 14, 6)</td>
<td>0</td>
</tr>
<tr>
<td>conv2d_1 (Conv2D)</td>
<td>(None, 10, 10, 16)</td>
<td>2416</td>
</tr>
<tr>
<td>max_pooling2d_1 (MaxPooling2D)</td>
<td>(None, 5, 5, 16)</td>
<td>0</td>
</tr>
<tr>
<td>flatten (Flatten)</td>
<td>(None, 400)</td>
<td>0</td>
</tr>
<tr>
<td>dense (Dense)</td>
<td>(None, 120)</td>
<td>48120</td>
</tr>
<tr>
<td>dense_1 (Dense)</td>
<td>(None, 25)</td>
<td>3025</td>
</tr>
<tr>
<td>dense_2 (Dense)</td>
<td>(None, 25)</td>
<td>650</td>
</tr>
<tr>
<td>Total params: 54,367</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trainable params: 54,367</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-trainable params: 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 30: Le-Net 5 Model Summary

{1, 32, 32, 1}
Predicted ASL Alphabet: Y
Results: ASL Alphabet Recognition

- The graph of the training accuracy was observed to be monotonically increasing with the number of epochs and accuracy was found to be 95.08% at the end of 20 epochs.
- Also, the loss was decreasing monotonically and reached to a final value of 0.2245
Experiments: NTU-RGB on LeNet-5 model

- When we trained the model on LeNet-5 architecture we observed 86% training accuracy for 3 classes whose gestures have complex motion in 3D space.
- The precision was best for eat gesture class which was most focused on depth dimension, i.e., Z coordinate.

```python
labels = np.argmax(yy, axis = 1)
# print(labels.shape, labels)
print(classification_report(y_test, labels, target_names =
    ['Hand Waving(Class 23)','Food Sign : Eat (Class 2)','Hello Sign (Salute) (Class 38)']))
```

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand Waving(Class 23)</td>
<td>0.86</td>
<td>0.83</td>
<td>0.84</td>
<td>270</td>
</tr>
<tr>
<td>Food Sign : Eat (Class 2)</td>
<td>0.89</td>
<td>0.88</td>
<td>0.89</td>
<td>301</td>
</tr>
<tr>
<td>Hello Sign (Salute) (Class 38)</td>
<td>0.82</td>
<td>0.85</td>
<td>0.83</td>
<td>279</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td>0.86</td>
<td>850</td>
</tr>
<tr>
<td>Macro Avg</td>
<td>0.86</td>
<td>0.85</td>
<td>0.85</td>
<td>850</td>
</tr>
<tr>
<td>Weighted Avg</td>
<td>0.86</td>
<td>0.86</td>
<td>0.86</td>
<td>850</td>
</tr>
</tbody>
</table>
Results: NTU-RGB+D on LeNet-5 Model

- As we can observe in the training accuracy and loss, the accuracy increased from 83.5% to 86% when learning rate was kept at 0.001 and Adam optimizer was used.
- The decrease in loss is also constant.
Experiments : NTU-RGB+D on SVM

- In this experiment we got accuracy of 89% for SVM model for the 3 classes.
- This is better as compared to the LeNet-5 model’s accuracy that was just 86%.
Experiment: Leap Motion Controller Classes Used

- ASL Actions:

  Please, Blue, Red, Yellow, Where, Stop, Water, Orange and Thanks performed majorly by right hand.
Experiments: Leap Motion Controller on SVM

• When we trained the model on this architecture, we got much better results than the LeNet-5 model.

• The overall training accuracy for SVM is 65% for 9 classes.

• We observed from the classification report that most classes got precision around 60% and “where” of the class was 100% classified.
Experiments: Leap Motion Controller on LeNet-5 model

• When we trained the model on this architecture, but it did not do well.
• We tried to use different learning rate like 0.00001, 0.001 and 0.0001 but the accuracy still was extremely poor around 13-15%.
• We also tried to change the activation function, epoch steps and number of epochs but accuracy was unaffected.
• As this dataset was totally focus on hand points, we deduced that the quality of dataset from the LMC for LeNet-5 as image classifier wasn’t good and hence it did not do well on this methodology proposed in this report.
Experiments : Impact on accuracy 2d vs 3d co-ordinates

- This experiment aimed at finding the performance of the model when only 2D (X, Y) data points of the skeleton are provided instead of 3D (X,Y, Z) using NTU RGB+D dataset.

- We observed that after removing the Z coordinate which maps to blue color in the RGB pallet, the temporal patterns remain same, but the dominance of colors is taken over by green and red with a huge contrast.

- Also, the accuracy wasn’t impacted much. There was a slight increase of 1% in the accuracy leading it to 90% overall training accuracy.

Figure 43: SVM Classification Report for 2D NTURGB Dataset
Results: Impact on accuracy 2d vs 3d co-ordinates

Figure 31: 3D v/s 2D temporal mapping for Class 2: Food (Eat)

Figure 32: 3D v/s 2D temporal mapping for Class 38: Hello (Salute)

Figure 33: 3D v/s 2D temporal mapping for Class 23: Bye

Output Temporal Images after transformation from Skeleton points to RGB image
Conclusions

- This project presented a prototype to help customers ask where to look for certain items in the store, seek suggestions for clothing or ask for navigating to a washroom or request for special assistance from a clerk.

- Accessibility technologies must provide real-time communication and be mandated in public places to aid those who are hard of hearing or deaf just like we have strict enforcements for websites.

- Also, with the spread of COVID-19 like virus, the demand for more and more contactless gesture-based technology is tremendously increasing.

- We can use these methodology and add more gesture to the dataset to make the system more user-friendly and robust. We can also try to use other neural networks for this methodology like LSTM that could possibly be used to get the real-time ASL speech.
References


