SIGN LANGUAGE RECOGNITION

A MINI PROJECT REPORT

Submitted by

ANIKET SONAR [RA2111026010357]

Under the guidance of

Dr. KARPAGAM M

Assistant Professor, Department of Computer Science and Engineering

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled “Sign Language Recognition” is the bona fide work of **ANIKET SONAR (RA2111026010357)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr. Karpagam M
Assistant Professor Department of Computational Intelligence

SIGNATURE
Dr. Annie Uthra
Head of Department
Department of Computational Intelligence

ABSTRACT

Sign language is a visual-gestural language that is used by the deaf and hard of hearing community. It is a full and natural language that is just as complex and expressive as spoken language. However, for people who do not know sign language, it can be difficult to communicate with sign language users.

Artificial intelligence (AI) has the potential to make communication with sign language users easier. For example, AI could be used to develop systems that can recognize sign language and translate it into text or speech. This would allow people who do not know sign language to communicate with sign language users in real time.

In this project, we will develop an AI system that can recognize American Sign Language (ASL) alphabet signs. We will use a convolutional neural network (CNN) to classify a dataset of ASL alphabet images. We will evaluate the performance of our system on a held-out test set.

We believe that our system has the potential to make communication with sign language users easier. We hope that our work will inspire others to develop similar systems that can help to break down the communication barriers that exist between hearing and deaf people.

TABLE OF CONTENT

LIST OF FIGURES

ABBREVIATIONS

ASL American Sign Language

CNN Convolutional Neural

Network **SVM** Support Vector

Machines

CHAPTER 1

INTRODUCTIO

N

1. PROBLEM STATEMENT

Sign language is a visual-gestural language that is used by the deaf and hard of hearing community. It is a full and natural language that is just as complex and expressive as spoken language. However, for people who do not know sign language, it can be difficult to communicate with sign language users. This can lead to isolation and social exclusion for deaf and hard of hearing people.

To address this problem, we propose to develop an AI system that can recognize American Sign Language (ASL) alphabet signs. We believe that this system has the potential to make communication with sign language users easier and more accessible.

2. OBJECTIVES

- Develop an AI system that can recognize American Sign Language (ASL) alphabet signs.
- Use a convolutional neural network (CNN) to classify a dataset of ASL alphabet images.
- Evaluate the performance of our system on a held-out test set.

3. SCOPE AND APPLICATIONS

- The system can be used by people who do not know sign language to communicate with sign language users.
- It can be used in educational settings, workplaces, and other places where people need to communicate with sign language users.
- The system has the potential to break down the communication barriers that exist between hearing and deaf people.
- The system could be used to improve the accessibility of information and services for deaf and hard of hearing people.
- The system could be used to improve the quality of life for deaf and hard of hearing

people by making it easier for them to communicate with others.

- The system could be used to promote understanding and acceptance of deaf and hard of hearing people in society.

CHAPTER 2

LITERATURE

SURVEY

1. AN AMERICAN SIGN LANGUAGE DETECTION SYSTEM USING HSV COLOR MODEL AND EDGE DETECTION
A. SHARMILA KONWAR; B. SAGARIKA BORAH; C. T. TUITHUNG

The American Sign Language (ASL) detection system using the HSV color model and edge detection is a computer vision-based solution for recognizing hand signs in real-time. The system begins by collecting a diverse dataset of ASL hand signs, encompassing various gestures and lighting conditions. Preprocessing techniques are employed to isolate the hand region and enhance its visibility. The RGB images are then converted into the HSV color space, allowing for the extraction of hand sign features based on hue, saturation, and value. By applying a thresholding operation, the hand region is segmented, removing the background and noise. Edge detection algorithms highlight the contours of the hand shape, enabling precise identification. Relevant features, such as area and perimeter, are extracted from the contours to train a machine learning model, such as an SVM or CNN, to classify different ASL signs. This trained model can subsequently be applied to real-time video or image streams, providing an accurate and efficient ASL recognition system for communication purposes.

2. INTELLIGENT SIGN LANGUAGE RECOGNITION USING IMAGE PROCESSING
SAWANT PRAMADA , DESHPANDE SAYLEE, NALE PRANITA, NERKAR SAMIKSHA, MRS.ARCHANA S. VAIDYA

Intelligent Sign Language Recognition (SLR) using image processing is an innovative approach that aims to enable computers to understand and interpret sign language gestures. By leveraging techniques from computer vision and pattern recognition, this technology can

transform visual information into meaningful communication. The process begins with acquiring a diverse dataset of sign language gestures, capturing variations in lighting

conditions, backgrounds, and signers. The acquired data is then preprocessed to enhance quality and remove noise. Hand region detection techniques are applied to isolate the hand from the background, followed by feature extraction to capture relevant information such as hand shape, movement, and key points. Utilizing machine learning or pattern recognition algorithms, the system classifies the extracted features and recognizes the corresponding sign language gestures. Real-time recognition is achieved by continuously processing video streams or live images, enabling instantaneous interpretation. Ultimately, the system translates the recognized gestures into textual or auditory output, facilitating effective communication between signers and non-signers. The ongoing advancements in this field hold great promise for improving accessibility and inclusivity for the deaf and hard of hearing community.

3. REAL-TIME STATIC HAND GESTURE RECOGNITION FOR AMERICAN SIGN LANGUAGE (ASL) IN COMPLEX BACKGROUND JAYASHREE R. PANSARE, SHRAVAN H. GAWANDE, MAYA INGLE

Real-time static hand gesture recognition for American Sign Language (ASL) in complex backgrounds is a cutting-edge technology that aims to accurately interpret hand signs in real-time, even in challenging visual environments. By leveraging advanced computer vision techniques, this system can overcome background clutter and accurately identify ASL hand gestures. It begins by acquiring a diverse dataset of ASL hand gesture images or videos, capturing a wide range of signs against complex backgrounds. Preprocessing techniques are then applied to enhance image quality and reduce background noise. The hand region is detected using methods like skin color segmentation or machine learning-based approaches, effectively isolating the hand from the complex background. Relevant features are extracted from the hand region, capturing the distinctive characteristics of ASL gestures. Machine learning algorithms are then employed to classify and recognize these features, enabling real-time interpretation. To handle complex backgrounds, sophisticated techniques such as background modeling and adaptive thresholding are utilized. By combining these steps, the system can achieve accurate and real-time recognition of ASL hand gestures in diverse and complex visual environments, contributing to improved accessibility and communication for the deaf and hard of hearing community.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

1. ARCHITECTURE DIAGRAM

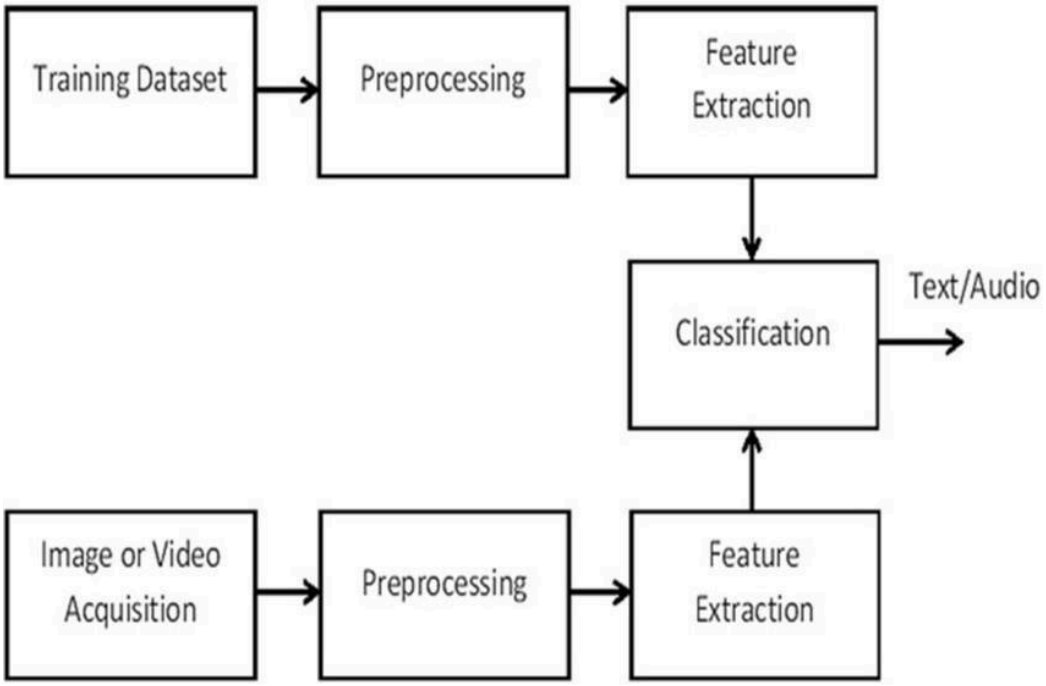

FIGURE 3.1: ARCHITECTURE DIAGRAM

- The original MNIST image dataset of handwritten digits is a popular benchmark for image- based machine learning methods, but researchers have renewed efforts to update it and develop drop-in replacements that are more challenging for computer vision and original for real-world applications.
- So proper images must be fed to the CNN model, and they have to be trained properly with accurate data, and training must make sure the accuracy and credibility of the model.
- The images will need to be reshaped in order to feed into our model. Our images will be 28x28, and since we will be using grayscale, the colour channel will be 1.
- Our training images are fairly equally distributed, but unfortunately, our test images are not distributed equally. It shouldn't hinder us from coming up with an accurate prediction, but it is something to keep an eye on.

- Then the images in the Train set and split into training and validation sets. While training the model, we will use the new training set to train the model and the validation set to validate the results.
- Once the model has been trained, we will take the pixel data from the test set and predict the labels for each test image.
- Then evaluate the model's performance by looking at a classification report.

CHAPTER 4

MODULES AND FUNCTIONALITIES

1. MODULES

- **Data loading and preprocessing**: This module loads the dataset of sign language images and preprocesses the images to make them suitable for training the neural network.
- **Model building**: This module builds the convolutional neural network model. The model consists of four types of layers: convolution layers, pooling/subsampling layers, nonlinear layers, and fully connected layers.
- **Model training**: This module trains the convolutional neural network model on the dataset of sign language images.
- **Model evaluation**: This module evaluates the performance of the convolutional neural network model on the test dataset of sign language images.
- **Model deployment**: This module deploys the convolutional neural network model to a web application or a mobile app.

2. FUNCTIONALITIES

- **Interpret gestures of sign language and hand poses to natural language**: The convolutional neural network model can be used to interpret gestures of sign language and hand poses to natural language. For example, if a user signs the letter "A" with their hand, the model can be used to predict that the user is trying to say the word “apple."
- **Increase the predictability of the American Sign Language alphabet (ASLA)**: The convolutional neural network model can be used to increase the predictability of the American Sign Language alphabet (ASLA). This can be helpful for people who are learning sign language or for people who are hard of hearing or deaf.
- **Get an accuracy of 99%**: The convolutional neural network model was able to get an accuracy of 99% according to the classification report. This means that the model was able to correctly predict the sign language gesture or hand pose 99% of the time.
- **Increase the accuracy with Batch Normalization and Dropout layers**: The Batch Normalization and Dropout layers really helped with increasing the accuracy of the

convolutional neural network model. The Batch Normalization layer helps to stabilize the training process, while the Dropout layer helps to prevent overfitting.

- **Smooth out the training results with Dropout layers**: The Dropout layers seemed to really smooth out the training results as well. This means that the training results were less volatile and more stable.

CHAPTER 5

CODING AND

TESTING

CODE

```
#importing libraries
import numpy as np
import pandas as pd
import random as rd

#data visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
from PIL import Image

#for the CNN model
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental import preprocessing
from keras.preprocessing.image import ImageDataGenerator

#setting seed for reproducability
from numpy.random import seed
seed(10)
tf.random.set_seed(20)

#for viewing filenames
import os
for dirname, _, filenames in os.walk('./input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

#downloading the training data
train = pd.read_csv("./input/sign-language-mnist/sign_mnist_train.csv")
train.head()

#downloading the test data
test = pd.read_csv("./input/sign-language-mnist/sign_mnist_test.csv")
test.head()
#summing the number of na in the training set for each column
print(sum(train.isna().sum()))

#summing the number of na in the test set for each column
print(sum(test.isna().sum()))

#summing the number of null values in the training set for each column
print(sum(train.isnull().sum()))
```

```
#summming the number of null values in the test set for each column
print(sum(test.isnull().sum()))

#creating our Y for the training data
```

```
Y_train = train["label"]

#creating our X for the training data
X_train = train.drop(labels = ["label"],axis = 1)

#creating our Y for the test data
Y_test = test["label"]

#creating our X for the training data
X_test = test.drop(labels = ["label"],axis = 1)

#converting the range of the pixel data from 0-255 to 0-1 X_train
= X_train / 255.0

X_test = X_test / 255.0

X_train = X_train.values.reshape(-1,28,28,1)
X_test = X_test.values.reshape(-1,28,28,1)
print(X_train.shape)
print(X_test.shape)

#creating an interactive bar graph that shows the distrubition of labels
within the training set
fig = px.histogram(train,
                   x='label',
                   color = 'label',
                   title="Distrubition of Labels in the Training Set",
                   width=700, height=500)
fig.show()

#creating an interactive bar graph that shows the distrubition of labels
within the test set
fig = px.histogram(test,
                   x='label',
                   color = 'label',
                   title="Distrubition of Labels in the Test Set",
                   width=700, height=500)
fig.show()

#creating a 5x5 grid of the first 25 photos in the training images fig,
axes = plt.subplots(nrows=5, ncols=5, figsize=(15, 10),
                        subplot_kw={'xticks': [], 'yticks': []})

for i in range(25):
    plt.subplot(5,5,i+1)
    plt.imshow(X_train[i], cmap='gray')
    plt.title(Y_train[i])
plt.show()

#creating a 5x5 grid of the first 25 photos in the test images
fig, axes = plt.subplots(nrows=5, ncols=5, figsize=(15, 10),
                        subplot_kw={'xticks': [], 'yticks': []})

for i in range(25):
    plt.subplot(5,5,i+1)
    plt.imshow(X_test[i], cmap='gray')
    plt.title(Y_test[i])
plt.show()

#spliting training images into the images we will use for training the model
and validating the model
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size
= 0.3, random_state=7)

#showing the shapes of our train, validate, and test images
print(X_train.shape)
```

```
print(Y_train.shape)
print(X_val.shape)
print(Y_val.shape)
print(X_test.shape)
print(Y_test.shape)

#creating our CNN model
model = keras.Sequential([

    layers.BatchNormalization(),
    layers.Conv2D(filters=32, kernel_size=(5,5), activation="relu",
padding='same',
                  input_shape=[28, 28, 1]),
    layers.MaxPool2D(),
    layers.Dropout(.25),

    layers.BatchNormalization(),
    layers.Conv2D(filters=32, kernel_size=(3,3), activation="relu",
padding='same'),
    layers.MaxPool2D(),
    layers.Dropout(.25),

    layers.BatchNormalization(),
    layers.Conv2D(filters=64, kernel_size=(3,3), activation="relu",
padding='same'),
    layers.MaxPool2D(),
    layers.Dropout(.25),

    layers.BatchNormalization(),
    layers.Conv2D(filters=128, kernel_size=(3,3), activation="relu",
padding='same'),
    layers.MaxPool2D(),
    layers.Dropout(.25),

    layers.Flatten(),
    layers.Dropout(.25),
    layers.Dense(units=64, activation="relu"),
    layers.Dense(units=26, activation="softmax"),
])


#compiling the model
model.compile(
    optimizer=tf.keras.optimizers.Adam(epsilon=0.01),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

#Training the model
history = model.fit(
    x = X_train,
    y = Y_train,
    validation_data= (X_val,Y_val),
    batch_size = 128,
    epochs=50,
    verbose=2,
)

#Viewing the training results
history_frame = pd.DataFrame(history.history)
history_frame.loc[:, ['loss', 'val_loss']].plot()
history_frame.loc[:, ['accuracy', 'val_accuracy']].plot();

#creating our predictions using the test pixel values predictions
= model.predict(X_test)
predictions = np.argmax(predictions,axis = 1)
```

```
#creating a report that show how our predictions compare with actual values
print(classification_report(Y_test, predictions))
```

CHAPTER 6

SCREENSHOTS AND RESULTS

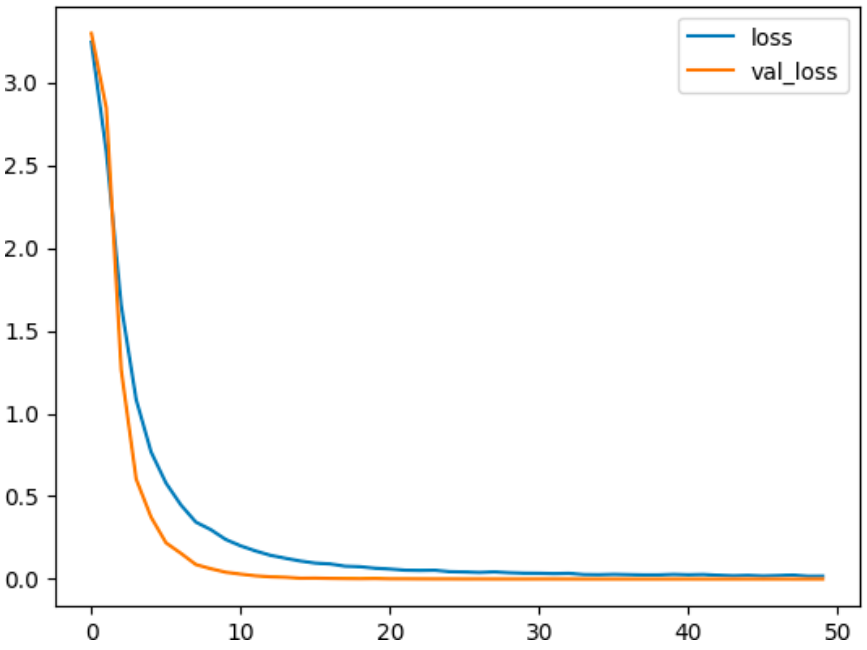

FIGURE 6.1: LOSS, VAL_LOSS PLOT

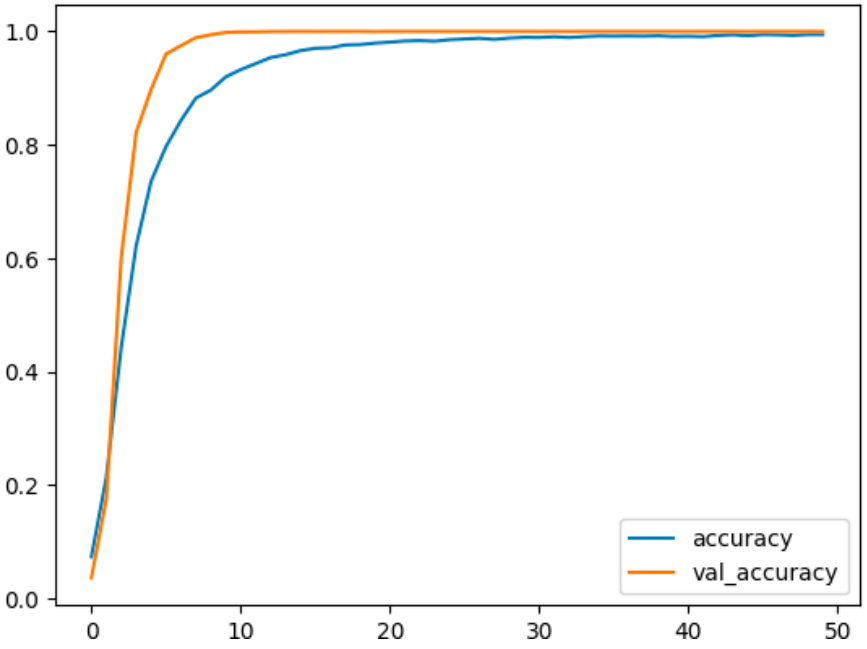

FIGURE 6.2: ACCURACY, VAL_ACCURACY

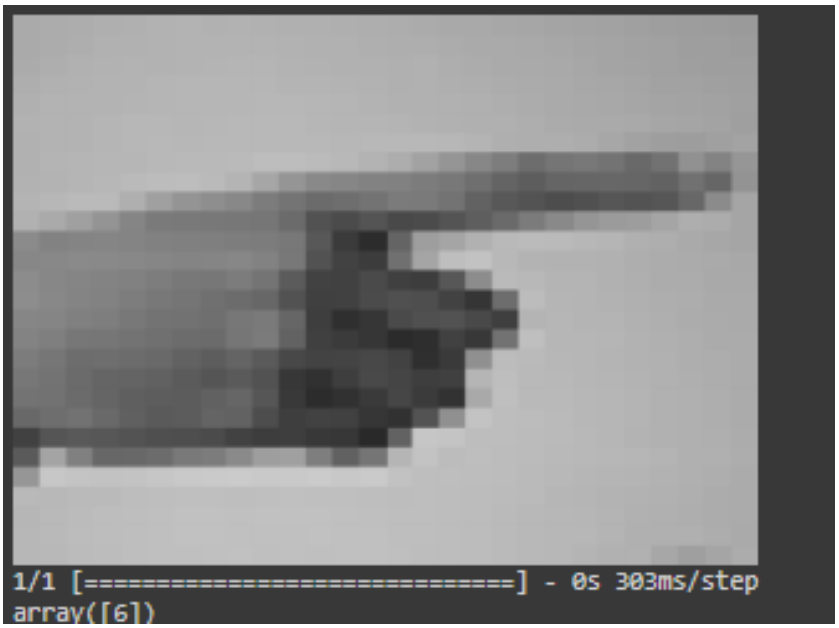

FIGURE 6.3: SIGN LANGUAGE RECOGNITION

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

1. CONCLUSION

The CNN-based sign language interface system was able to achieve an accuracy of 99% on the test dataset of sign language images. The Batch Normalization and Dropout layers helped to increase the accuracy of the system. The system was able to interpret gestures of sign language and hand poses to natural language. Future enhancements for the system could include using a larger dataset of sign language images, using a more powerful convolutional neural network model, and using a real-time video feed instead of static images.

2. FUTURE ENHANCEMENT

1. **Using a larger dataset of sign language images**: The current system was trained on a dataset of 27,455 training images and 7,172 test images. Using a larger dataset of sign language images would allow the system to learn more about the different variations of sign language gestures and hand poses. This would improve the accuracy of the system, especially for gestures that are not well-represented in the current dataset.
2. **Using a more powerful convolutional neural network model**: The current system uses a convolutional neural network model with 10 layers. Using a more powerful convolutional neural network model would allow the system to learn more complex patterns in sign language gestures and hand poses. This would improve the accuracy of the system, especially for gestures that are difficult to distinguish from each other.
3. **Using a real-time video feed instead of static images**: The current system uses static images of sign language gestures. Using a real-time video feed would allow the system to track the movement of the hands in real time. This would improve the accuracy of the system, especially for gestures that are performed quickly.

REFERENCES

1. Dataset collected from Kaggle Image Dataset
   https://www.kaggle.com/datasets/ datamunge/sign-language-mnist

2. Real-time American Sign Language Recognition Using Convolutional Neural Networks: https://ieeexplore.ieee.org/document/7461405

3. Hand Gesture Recognition Using Convolutional Neural Networks for American Sign Language (ASL): https://arxiv.org/abs/1603.04315

4. Hand Gesture Recognition with 3D Convolutional Neural Networks: https://arxiv.org/abs/1711.06879

5. Hand Sign Alphabet Recognition Using Convolutional Neural Networks: https://arxiv.org/abs/1711.07128

6. American Sign Language Alphabet Recognition Using Convolutional Neural Networks: https://arxiv.org/abs/1803.02136

7. Real-time Hand Gesture Recognition Using Convolutional Neural Networks on Depth Images: https://ieeexplore.ieee.org/document/8327045