Project Report

On

Sign Language Recognition

at

U.V. Patel College of Engineering

Prepared By:

Mr.Yagnesh Pansuriya(18012011054)

Guided By:

Prof.Ketan Sarvakar

B.tech Semester-VII

(Computer Engineering)

Submitted to,

Department of Computer Engineering

U.V. Patel College of Engineering

Ganpat University, Kherva-384012

1. Abstract

Sign language is used by deaf and hard hearing people to exchange information between their own community and with other people. Computer recognition of sign language deals from sign gesture acquisition and continues till text/speech generation. Sign gestures can be classified as static and dynamic. However static gesture recognition is simpler than dynamic gesture recognition but both recognition systems are important to the human community. The sign language recognition steps are described in this survey. The data acquisition, data preprocessing and transformation, feature extraction, classification and results obtained are examined.

2. Introduction

Sign language (SL) is a visual-gestural language used by deaf and hard-hearing people for communication purposes. Three dimensional spaces and the hand movements are used (and other parts of the body) to convey meanings. It has its own vocabulary and syntax which is purely different from spoken languages/written language. Spoken languages use the oratory faculties to produce sounds mapped against specific words and grammatical combinations to convey meaningful information. Then the oratory elements are received by the auditory faculties and processed accordingly. Sign language uses the visual faculties which is different from spoken language. Spoken language makes use of rules to produce comprehensive messages; similarly sign language is also governed by a complex grammar. A sign language recognition system consists of an easy, efficient and accurate mechanism to transform sign language into text or speech. The computerized digital image processing and a wide variety of classification methods used to recognize the alphabet flow and interpret sign language words and phrases. Sign language information can be conveyed using gestures of hands. Three essential components in a gesture recognition system are: gesture modeling, gesture analysis and gesture recognition.

Fig.2.1-Different signs

3. Tools and technologies

Tools and libraries	Description
Spyder	Spyder is a free and open source scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.
CNN	In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.
Keras	Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.
Numpy	NumPy is the fundamental package for scientific computing in Python. ... NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed

	more efficiently and with less code than is possible using Python's built-in sequences.

4. Architecture of CNN

There are two main parts to a CNN architecture:

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

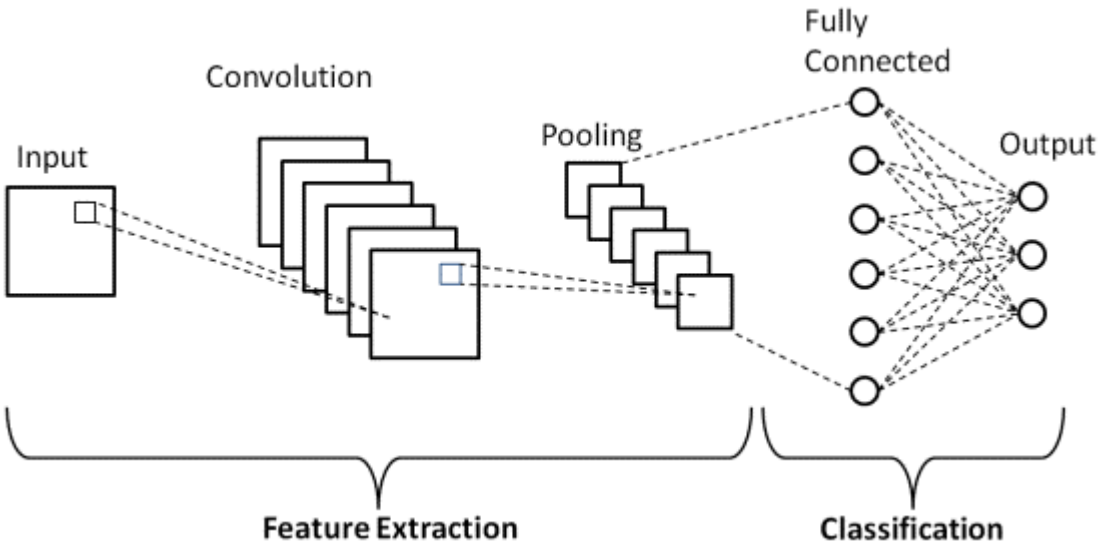


Fig.4.1-CNN architecture

Convolution Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers

and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.

Fully Connected Layer
The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

Dropout
Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data.

Activation functions
Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.

5. **Dataset**
The dataset format is patterned to match closely with the classic MNIST. Each training and test case represents a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions). The training data (27,455 cases) and test data (7172 cases) are approximately half the size of the standard MNIST but otherwise similar with a header row of label, pixel1,pixel2….pixel784 which represent a single 28x28 pixel image with grayscale values between 0-255.
Access Link: https://www.kaggle.com/datamunge/sign-language-mnist

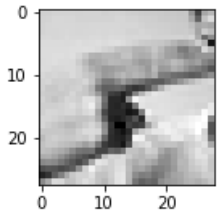
6. **Code**
Code is available at github.
https://github.com/yagnesh-collab/Sign-language-recognization

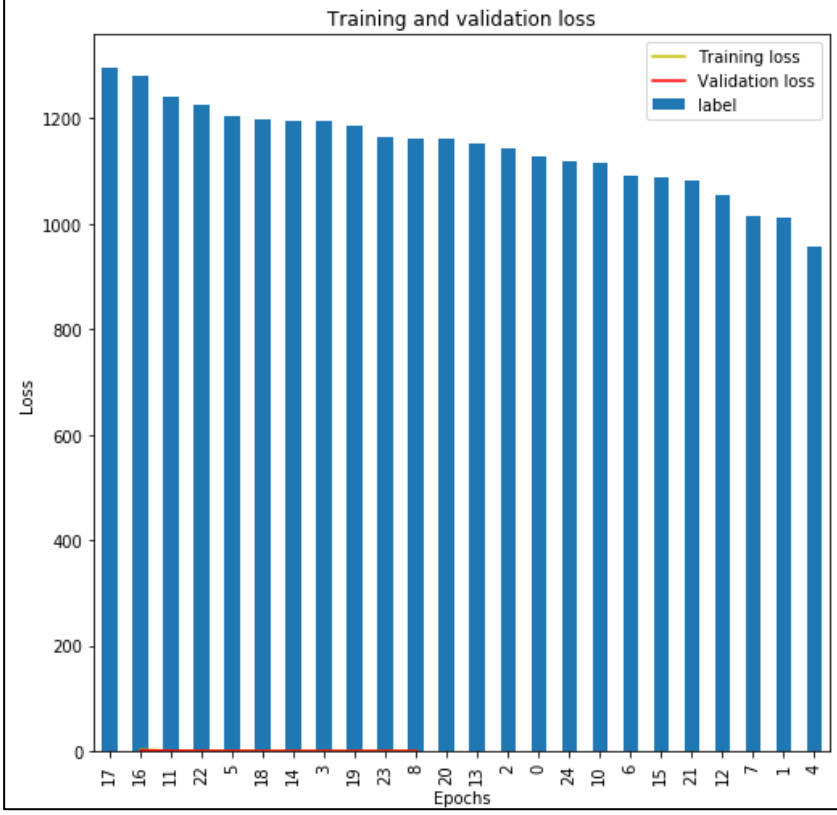
7. **Screenshots**

```
In [1]: runfile('C:/Users/yagnesh/.spyder-py3/chatbot.py', wdir='C:/Users/yagnesh/.spyder-py3')
Label for the image is:  H
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1, 1, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 128)               16512
```

```
_________________________________________________________________
Epoch 1/10
215/215 [==============================] - 22s 96ms/step - loss: 2.9756 - acc: 0.1074 - val_loss: 1.3161 -
val_acc: 0.5943
Epoch 2/10
215/215 [==============================] - 20s 94ms/step - loss: 1.1688 - acc: 0.6089 - val_loss: 0.6922 -
val_acc: 0.7521
Epoch 3/10
215/215 [==============================] - 24s 110ms/step - loss: 0.6350 - acc: 0.7847 - val_loss: 0.4652 -
val_acc: 0.8415
Epoch 4/10
215/215 [==============================] - 25s 117ms/step - loss: 0.3943 - acc: 0.8658 - val_loss: 0.3197 -
val_acc: 0.8961
Epoch 5/10
215/215 [==============================] - 29s 137ms/step - loss: 0.2704 - acc: 0.9108 - val_loss: 0.2586 -
val_acc: 0.9070
Epoch 6/10
215/215 [==============================] - 22s 102ms/step - loss: 0.1965 - acc: 0.9362 - val_loss: 0.2235 -
val_acc: 0.9334
Epoch 7/10
215/215 [==============================] - 20s 95ms/step - loss: 0.1506 - acc: 0.9525 - val_loss: 0.2047 -
val_acc: 0.9338
Epoch 8/10
215/215 [==============================] - 20s 91ms/step - loss: 0.1168 - acc: 0.9636 - val_loss: 0.2140 -
val_acc: 0.9373
Epoch 9/10
215/215 [==============================] - 19s 89ms/step - loss: 0.0910 - acc: 0.9711 - val_loss: 0.2132 -
val_acc: 0.9412
Epoch 10/10
215/215 [==============================] - 20s 91ms/step - loss: 0.0773 - acc: 0.9751 - val_loss: 0.1778 -
val_acc: 0.9513
```

```
val_acc: 0.9513
```






```
C:\anaconda\lib\site-packages\tensorflow\python\keras\engine\sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01. Please use instead:*
`np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses a
`softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary
classification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
Accuracy Score =  0.9513385387618516
Predicted Label:  E
True Label:  E
```