2022-23

YESHWANTRAO CHAVAN COLLEGE OF ENGINEERING, NAGPUR.

(An autonomous Institution Affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)

DEPARTMENT OF COMPUTER TECHNOLOGY

Hand Sign Recognition Using Machine Learning

NAME OF THE STUDENT: Ayush Raut, Nikhil Dapkekar, Piyush Dabare, Jay Rathod

Abstract: There is an undeniable communication problem between the deaf community and the hearing majority which can be solved by using innovation ideas and techniques which can be used to automatically recognize sign language and convert it into text. Sign Language is used by the deaf and voiceless community to be able to communicate with others, but the most commonly faced problem here is that everyone around may not be able to understand sign language. Written communication can be also be used but it is slow as compared to face-to-face communication. In case emergency situations its more efficient to use sign language than written communication therefore, many sign languages were created in different countries of world. To solve this problem, we can use a custom Convolution Neural Networks (CNN) model to recognize hand gestures.

Introduction:

Sign Language is used by the deaf and voiceless community. It is a type of hand gesture-based language which is used to overcome the barrier caused due to difficulties from hearing issues.

Convolution Neural Networks (CNN): Convolution Neural Networks (CNN) are regularized versions of multilayer perceptron's. Multilayer perceptron's usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. For that we need to implement a convolutional layer, pooling layer, and fully connected layer. Convolution layer performs a dot product between two matrices, where one matrix is a filter and the other matrix is the restricted portion of the image. Pooling layer is applied to decrease the computational power required to process the data. It is done by decreasing the dimensions of the featured matrix even more.

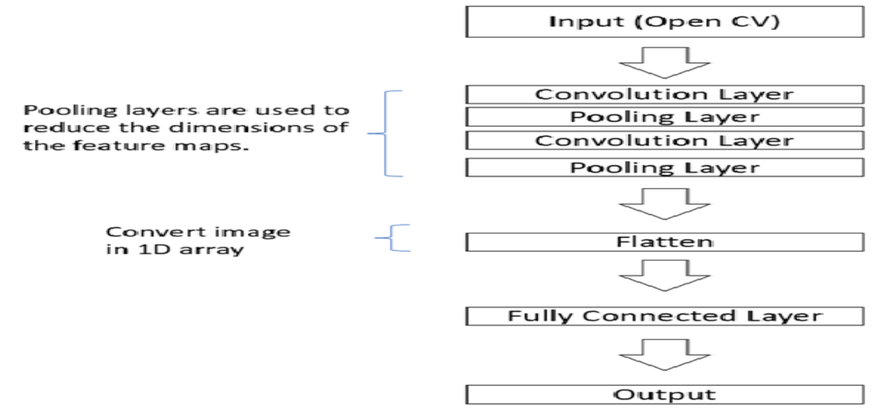
Simulation Flow Diagram:

In **convolution layer** we take a small window size [typically of length 5*5] that extends to the depth of the input matrix.

We use **pooling layer** to decrease the size of activation matrix and ultimately reduce the learnable parameters.

In a **fully connected region**, we will connect all the inputs to neurons.

After getting values from fully connected layer, we will connect them to the final layer of neurons [having count equal to total number of classes], that will predict the probability of each image to be in different classes.



Simulation Result:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 32)      320

 max_pooling2d (MaxPooling2D  (None, 64, 64, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 64, 64, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 32, 32, 32)       0
 2D)

 flatten (Flatten)           (None, 32768)             0

 dense (Dense)               (None, 128)               4194432

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 96)                12384

 dropout_1 (Dropout)         (None, 96)                0

 dense_2 (Dense)             (None, 64)                6208

 dense_3 (Dense)             (None, 27)                1755

=================================================================
Total params: 4,224,347
```

```
classifier.fit(training_set,
               epochs = 5,
               validation_data = test_set)
```

```
Epoch 1/5
1285/1285 [==============================] - 1127s 874ms/step - loss: 1.9926 - accuracy: 0.3847 - val_loss: 0.4755 - val_accuracy: 0.8299
Epoch 2/5
1285/1285 [==============================] - 466s 362ms/step - loss: 0.8373 - accuracy: 0.7181 - val_loss: 0.1501 - val_accuracy: 0.9545
Epoch 3/5
1285/1285 [==============================] - 216s 168ms/step - loss: 0.5502 - accuracy: 0.8175 - val_loss: 0.0651 - val_accuracy: 0.9843
Epoch 4/5
1285/1285 [==============================] - 343s 267ms/step - loss: 0.4141 - accuracy: 0.8662 - val_loss: 0.0289 - val_accuracy: 0.9902
Epoch 5/5
1285/1285 [==============================] - 426s 332ms/step - loss: 0.3314 - accuracy: 0.8940 - val_loss: 0.0175 - val_accuracy: 0.9963

<keras.callbacks.History at 0x1522fd4e500>
```

Simulation Output:

Conclusion and Future scope:

Conclusion:

Sign Language recognition (SLR) is very complex topic because of the complexity and diversity of Sign Languages. Sign language recognition is a collaborative research field that includes natural language processing, computer vision, pattern matching, and linguistics. There are many different types of Sign Language used in region of world. For example, Indian sign language (ISL) uses two hand to symbolize letter "a" while American Sign Language (ASL) required only one hand. We can build UI so that it can become user friendly to use.

Future scope:

- The aim of this project is to build a real time SLR computer application that can automatically detect hand gestures in natural lighting conditions and convert that into text.
- The model can be further trained with a dataset such that its accuracy does not get affected by the background of image gesture which is taken from camera.
- Modules can be optimized and integrated in AI systems which are used by general public so that it can available to use. For example, Google assistant.

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	3	3	3	3	3		3				2	3		2
CO2	3		3	3	3	3	3	3	3	3	2	3	3	3
CO3						3				3	2	3		
AVG	3	3	3	3	3	3	3	3	3	3	2	3	3	2.5

Signature of Guide