Project Report of Internship Program 2021

On

**“SIGN LANGUAGE RECOGNITION”**

**MEDTOUREASY**



26th April 2021

**Contents**

[**1.ABOUT THE COMPANY** 3](#_Toc70188907)

[**2.ABSTRACT** 4](#_Toc70188908)

[**3.METHODOLOGY** 5](#_Toc70188909)

[**4.IMPLEMENTATION** 6](#_Toc70188910)

[**4.1.American Sign Language (ASL)** 6](#_Toc70188911)

[**4.2.Visualize the training data** 6](#_Toc70188912)

[**4.3.Examine the Dataset** 7](#_Toc70188913)

[**4.4.One-Hot Encode the Data** 8](#_Toc70188914)

[**4.5.Define the Model** 9](#_Toc70188915)

[**4.6.Compile the model** 11](#_Toc70188916)

[**4.7.Train the model** 12](#_Toc70188917)

[**4.8.Test the Model** 13](#_Toc70188918)

[**4.9.Visualize Mistakes** 14](#_Toc70188919)

[**5.CONCLUSION** 18](#_Toc70188920)

[**6.FUTURE SCOPE** 18](#_Toc70188921)

[**7.REFERENCES** 19](#_Toc70188922)

# 

# **1.ABOUT THE COMPANY**

MedTourEasy, a global healthcare company, provides you the informational resources needed to evaluate your global options. It helps you find the right healthcare solution based on specific health needs, affordable care while meeting the quality standards that you expect to have in healthcare.

MedTourEasy improves access to healthcare for people everywhere. It is an easy-to-use platform and service that helps patients to get medical second opinions and to schedule affordable, high-quality medical treatment abroad.

# 

# **2.ABSTRACT**

American Sign Language (ASL) is the primary language used by many deaf individuals in North America, and it is also used by hard-of-hearing and hearing individuals. The language is as rich as spoken languages and employs signs made with the hand, along with facial gestures and bodily postures.

ASL originated in the early 19th century in the American School for the Deaf (ASD) in West Hartford, Connecticut, from a situation of language contact. Since then, ASL use has propagated widely by schools for the deaf and Deaf community organizations. Despite its wide use, no accurate count of ASL users has been taken. Reliable estimates for American ASL users range from 250,000 to 500,000 persons, including a number of children of deaf adults.

A lot of recent progress has been made towards developing computer vision systems that translate sign language to spoken language. This technology often relies on complex neural network architectures that can detect subtle patterns in streaming video. However, as a first step, towards understanding how to build a translation system, we can reduce the size of the problem by translating individual letters, instead of sentences.



# **3.METHODOLOGY**

The steps that are followed to train a convolutional neural network to classify ASL letters are as follows:

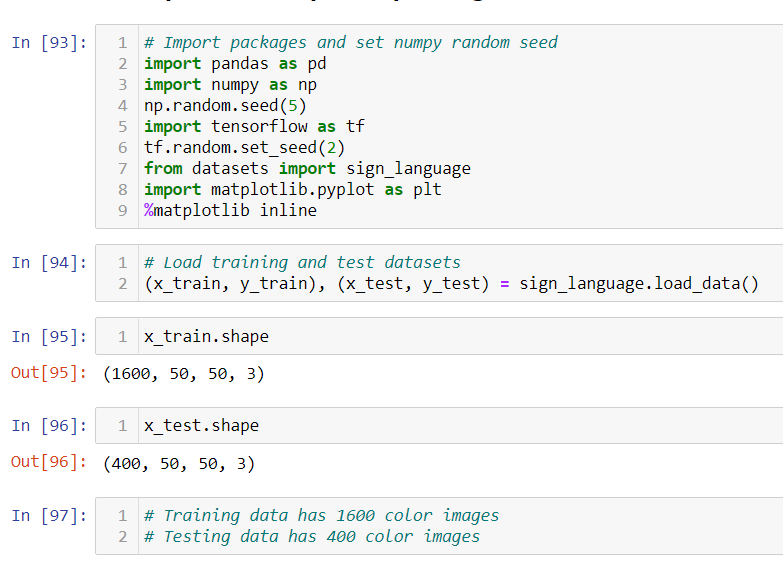
* Importing the packages and loading the data
* Visualize the training data.
* Examine the dataset
* One-hot encode the data
* Define the model
* Compile the model
* Test the model
* Visualize mistakes

Each of these steps is explained in detail in the following sections.

This project is done on Jupyter notebook (IDE) and in Python (Programming Language). Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Python is an interpreted high-level general-purpose programming language.

# **4.IMPLEMENTATION**

## **4.1. Importing the packages and loading the data**



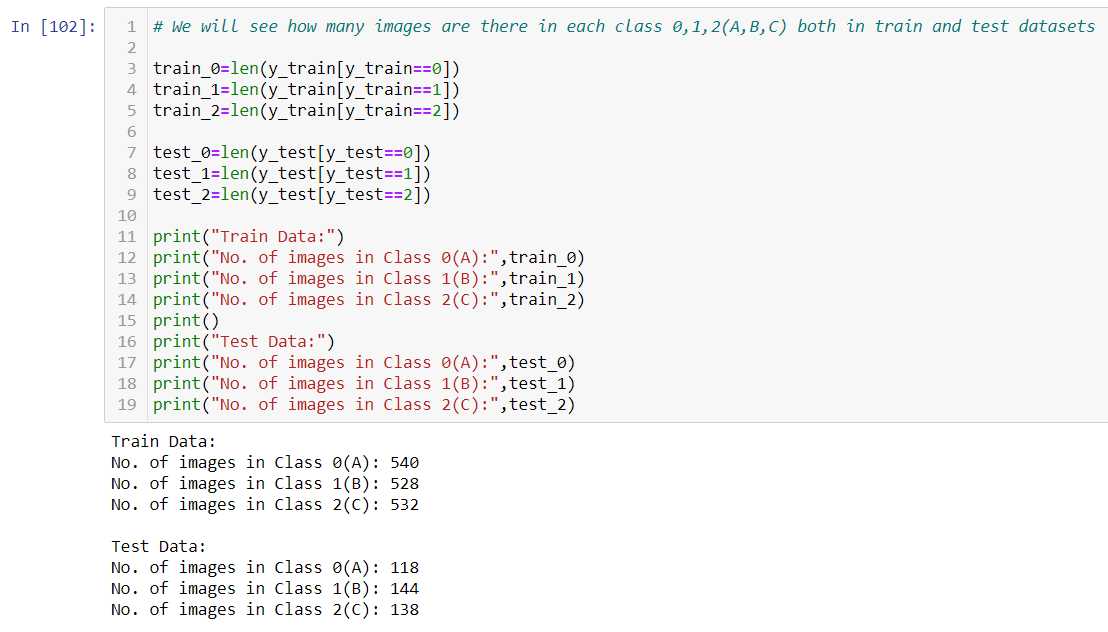
## **4.2. Visualize the training data**

American Sign Language is an image dataset that contains images related to sign of each of letters. The ASL dataset contains total of 26 signs corresponding to each English alphabet. Here we have taken only 3 signs and have assigned ‘A’,’B’,’C’ to labels to visualize the training data. The output is as shown.



## **4.3. Examine the Dataset**

Once the dataset is visualized, we examine the dataset to see how many images of each letter can be found in the dataset. The dataset is already split into x\_train, y\_train, x\_test and y\_test. Entry in y\_train, y\_test is either 0,1 or 2 corresponding to ‘A’,’B’,’C’.



Here we see that in the training set there are 540 images corresponding to A, 528 images of B and 532 images of C.

In the Test set there are 118 images of A,144 images corresponding to B and 138 images of C.

## 

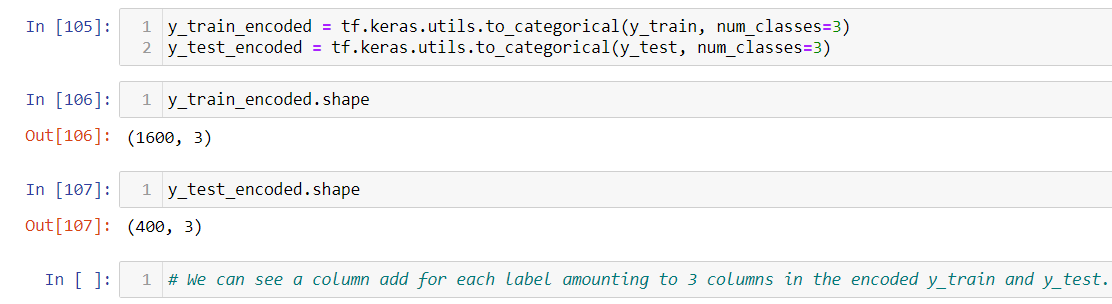
## **4.4. One-Hot Encode the Data**

Need for one-hot encoding: Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric.

For categorical variables where no ordinal relationship exists, one-hot encoding can be done to convert categorical variables to numeric.

* Here, 0 is encoded as [1, 0, 0],
* 1 is encoded as [0, 1, 0], and
* 2 is encoded as [0, 0, 1].

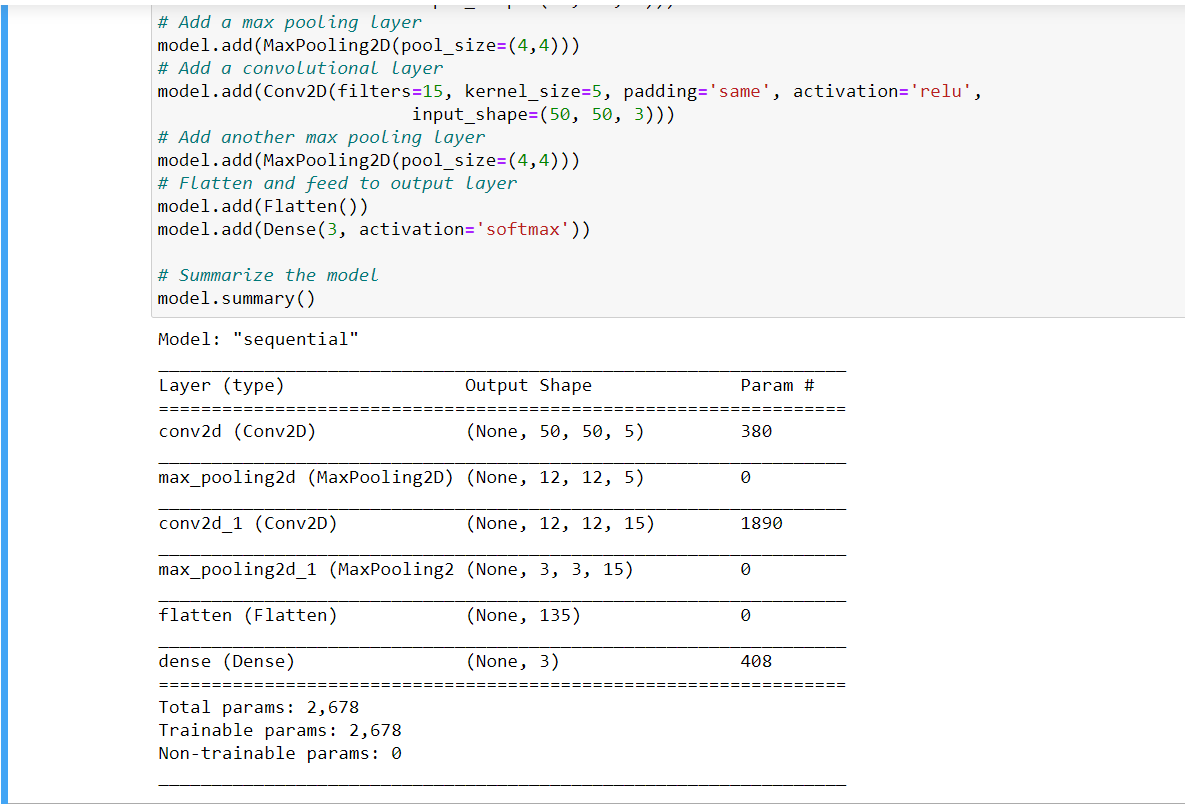
To do so we can use Keras built-in function to\_categorical() as shown .



## **4.5. Define the Model**

We now define a convolutional neural network to classify data with help of functions such as Conv2D, MaxPooling2D, Flatten and Dense.

* **Sequential** is used to initialize the neural network.
* **Convolution2D** is used to make the convolutional network that deals with the images. The network is specified with the following arguments.
* **filters**: Integer, the dimensionality of the output space (i.e., the number of output filters in the convolution).
* **kernel size**: An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
* **padding**: one of "valid" or "same" (case-insensitive). "valid" means no padding. "same" results in padding evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input.
* **activation**: Activation function to use. If you don't specify anything, no activation is applied.
* **MaxPooling2D** layer is used to add the pooling layers. **pool\_size**: integer or tuple of 2 integers, window size over which to take the maximum. (2, 2) will take the max value over a 2x2 pooling window. If only one integer is specified, the same window length will be used for both dimensions.
* **Flatten** is the function that converts the pooled feature map to a single column that is passed to the fully connected layer.
* **Dense** adds the fully connected layer to the neural network.



This is the output obtained after defining the model. We can see the output Shape and number of parameters corresponding to each of the functions conv2d, max\_pooling etc.

We also obtain the output of Total parameters and number of trainable and non-trainable parameters after defining the model.

## 

## **4.6. Compile the model**

We compile the above defined model with ‘adam’ optimizer.

The purpose of loss functions is to compute the quantity that a model should seek to minimize during training.

categorical\_crossentropy computes the crossentropy loss between the labels and predictions.

The metric accuracy calculates how often predictions are equal to labels.

Optional arguments of accuracy include:

* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.



## **4.7. Train the model**

After compiling we train the model with the help of. fit() method.

.fit() method primarily takes two arguments

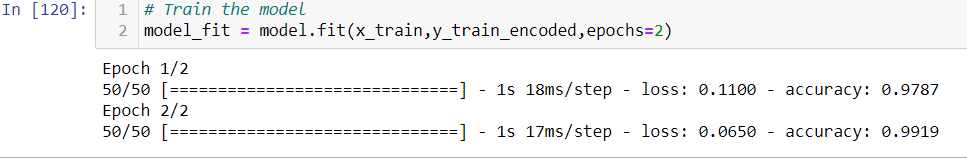
1.X: Input data (Training set’s independent labels) and

2.y: Target data (Training set’s target variable).

Along with these two it can also take arguments such as :

* batch\_size: Integer or None. Number of samples per gradient update. If unspecified, batch\_size will default to 32.
* epochs: Integer. Number of epochs to train the model. An epoch is an iteration over the entire x and y data provided.
* validation\_split: Data on which to evaluate the loss and any model metrics at the end of each epoch. The model will not be trained on this data.

Verbose, callbacks, shuffle, class\_weight etc are some of the other arguments that. fit() method can take.

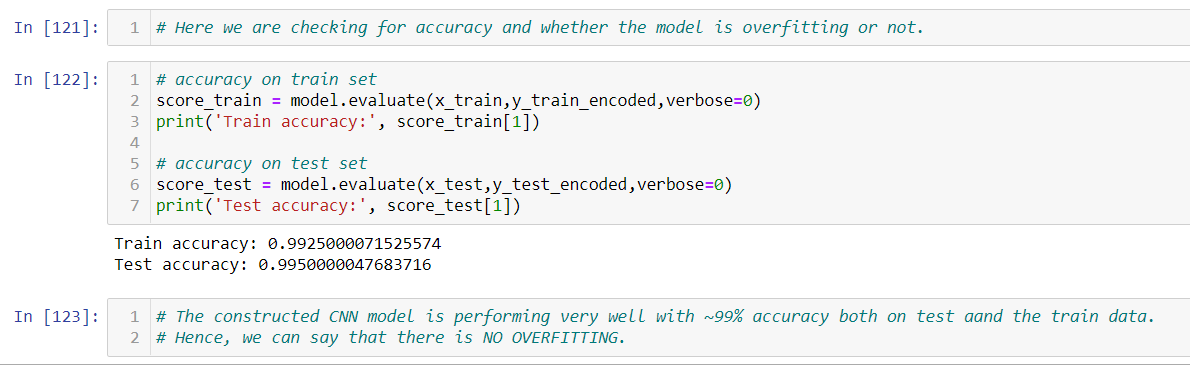


## **4.8. Test the Model**

After training the model we test the model with the help of “.**evaluate()**” method.

Evaluation is a process during development of the model to check whether the model is best fit for the given problem and corresponding data. Keras model provides a function, evaluate which does the evaluation of the model. It has three main arguments,

* Test data
* Test data label
* verbose - true or false



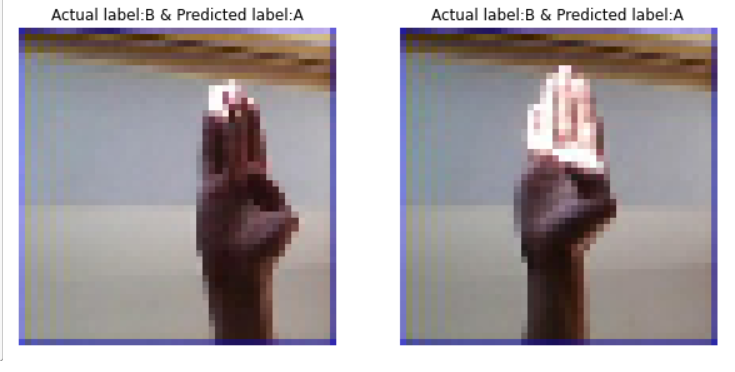
The model is a good fit with an accuracy of ~99% on both the train and test datasets. Hence, we can say that there is no overfitting on the training set.

## 

## **4.9. Visualize Mistakes**

Here we Visualize images that were incorrectly classified by the model.

Our model classified with a good accuracy of 99% and only 2 images were misclassified.



# 

# **5.CONCLUSION**

# 1. We divided the data into training and test sets with 1600 and 400 images respectively.

# 2. Visualized the training data and familiarized with the sign symbols.

# 3. Labels 'A', 'B' and 'C' are encoded as 0, 1, and 2, respectively for both train and test datasets.

# 4. Defined a convolutional neural network to classify the data.

# 5. Compiled the model with the 'adam' optimizer, 'categorical\_crossentropy' as the loss function, and 'accuracy' as a metric.

# 6. Trained the data with the arguments x\_train,y\_train\_encoded,epochs=2.

# 7. The model is a good fit with an accuracy of ~99% on both the train and test datasets.

# 8. Visualised the model prediction mistakes which were only 2 misclassifications.

# **6.FUTURE SCOPE**

Only 3 sign languages were used in this dataset, further scope can be collecting images for other signs and to build the model. Also, Neural networks can be built to translate words and sentences of sign language.

# **7.REFERENCES**

a.[Image Processing in Python | DataCamp](https://learn.datacamp.com/courses/image-processing-in-python)

b.[Python (programming language) - Wikipedia](https://en.wikipedia.org/wiki/Python_%28programming_language%29)

c.[Deep Learning with Keras - Training the Model - Tutorialspoint](https://www.tutorialspoint.com/deep_learning_with_keras/deep_learning_with_keras_training_the_model.htm)

d.<https://keras.io/>