Final Report of Traineeship Program 2023

The Training opportunity that I had with MedTourEasy was a great chance for learning and understanding the intricacies of the subject of Deep Learning; and also, for personal as well as professional development. I am very obliged for having a chance to interact with so many professionals who guided me throughout the internship project and made it a great learning curve for me.

Firstly, I express my sincere gratitude and special thanks to the Training & Development Team of MedTourEasy who allowed me to complete my internship at their esteemed organization. Also, I thank the team for making me understand the details of the Machine Learning profile and training me in the same so that I can carry out the project properly and with maximum client satisfaction and also for spearing his valuable time despite his busy schedule.

I would also like to thank the team of MedTourEasy and my colleagues who made the working environment productive and very conducive.

By,
Jenina Angelin D
Deep Learning Trainee
MedTourEasy

GitHub Link: https://github.com/JeninaAngelin/Sign-Language-Recognition

ABSTRACT

The communication gap between the deaf and hearing population is clearly noticed. To make possible communication between the Deaf and the hearing population and to overpass the gap in access to next-generation Human-Computer Interfaces, automated sign language analysis is highly crucial. Exploration and experimentation of an efficient methodology based on facet features analysis. For a recognition system that can recognize gestures from video which can be used as a translation, A methodology has been proposed that extracts candidate hand gestures from a sequence of video frames and collect hand features. The system has three separate parts namely: Hand Detection, Shape Matching, and Hu moments comparison. The Hand Detection section detects hands through skin detection and by finding contours. It also includes the processing of video frames. The procedure of shape matching is attained by comparing the histograms. The values of Hu moments of the candidate hand region is identified using contour region analysis and compared to run matches and identify the particular sign language alphabet. The experimental analysis supports the efficiency of the proposed methodology on benchmark data.

American Sign Language (ASL) is a visual-gestural language that is used by many people who are deaf or hard of hearing. In this paper, we design a visual recognition system based on action recognition techniques to recognize individual ASL signs. Specifically, we focus on the recognition of words in videos of continuous ASL signing. The proposed framework combines multiple signal modalities because ASL includes gestures of both hands, body movements, and facial expressions. We have collected a corpus of RBG + depth videos of multi-sentence ASL performances, from both fluent signers and ASL students; this corpus has served as a source for training and testing sets for multiple evaluation experiments reported in this paper. Experimental results demonstrate that the proposed framework can automatically recognize ASL.

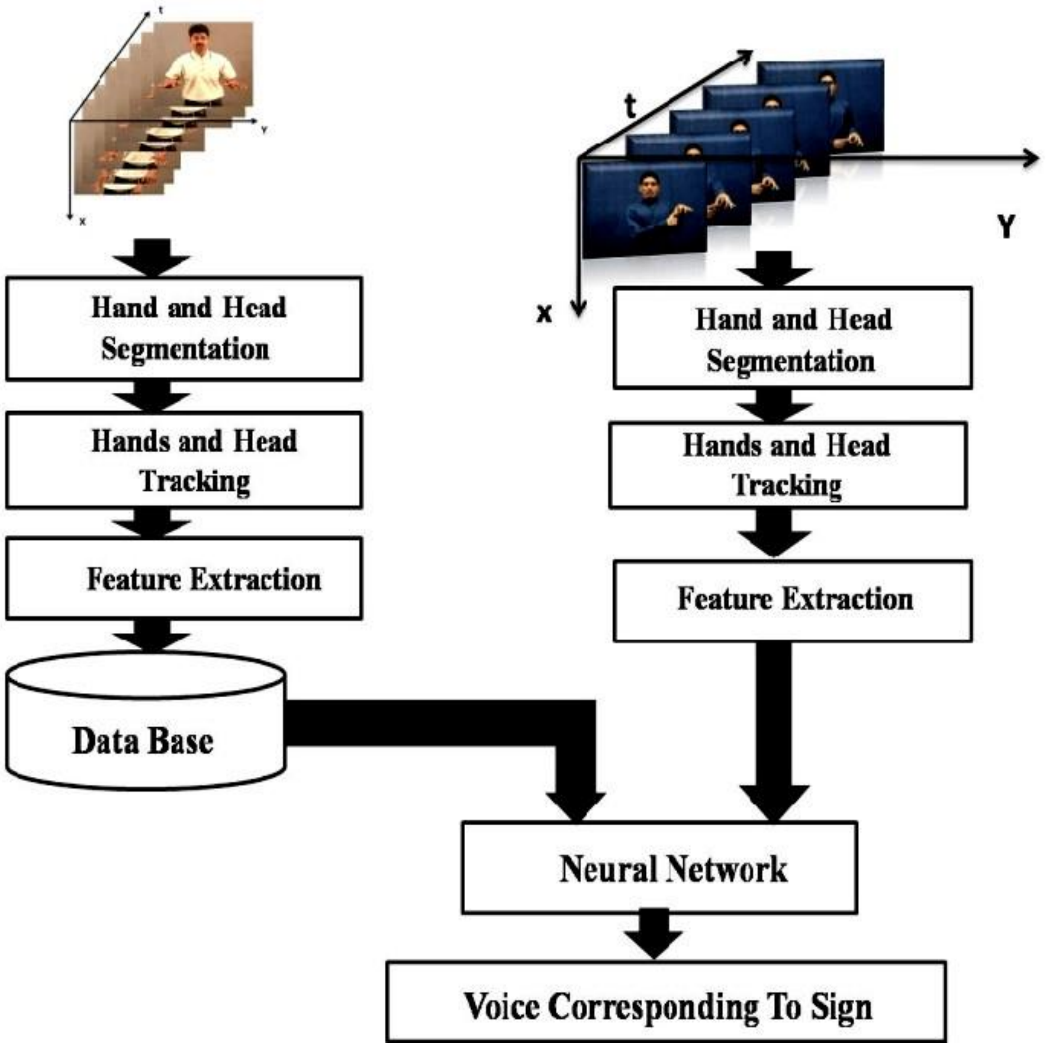
About the Project

The National Institute on Deafness and Other Communications Disorders (NIDCD) indicates that the 200-year-old American Sign Language is a complete, complex language (of which letter gestures are only part) but is the primary language for many deaf North Americans.

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.

Therefore, building a system that can recognize sign language will help the deaf and hard of hearing better communicate using modern-day technologies. In this article, we will go through different architectures of CNN and see how it performs in classifying Sign Language.

Use Case Diagram



The above figure shows the use case of the project. UML illustrates software both behaviorally (dynamic view) and structurally (static view). It has a graphical notation to create visual models of the systems and permits extension with a profile (UML). The UML includes elements Such as: Actors, Activities, Use Cases, Components, and so on. The Use Case diagram is a behavioral UML diagram. It expresses the functionality offered by a system in terms of actors, their objectives characterize as use cases, and any reliance amongst these use cases.

Implementation

Data Collection

The primary source of data for this project was the compiled dataset of American Sign Language (ASL) called the ASL Alphabet from Kaggle user Akash [3]. The dataset is comprised of 87,000 images which are 200x200 pixels. There are 29 total classes, each with 3000 images, 26 for the letters A-Z and 3 for space, delete, and nothing. This data is solely of the user Akash gesturing in ASL, with the images taken from his laptop's webcam. These photos were then cropped, rescaled, and labeled for use.

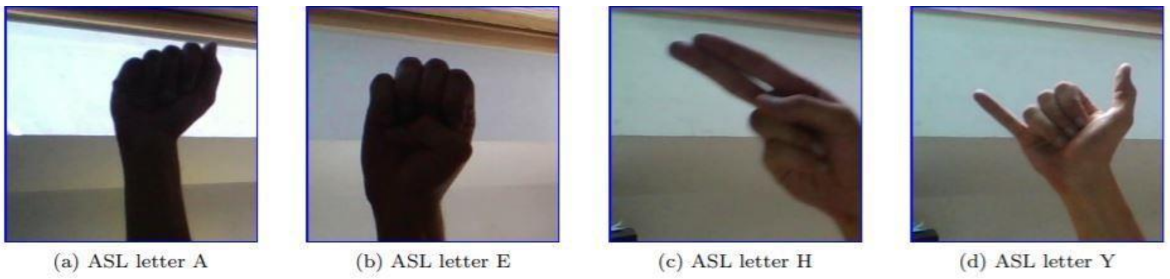
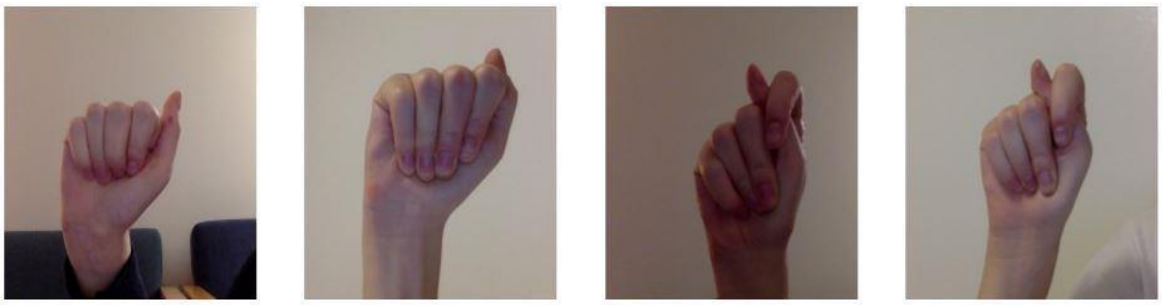


Figure 2: Examples of images from the Kaggle dataset used for training. Note difficulty of distinguishing fingers in the letter E.

A self-generated test set was created in order to to investigate the neural network's ability to generalize. Five different test sets of images were taken with a webcam under different lighting conditions, backgrounds, and use of dominant/non-dominant hand. These images were then cropped and preprocessed.

Data Pre-processing

The data preprocessing was done using the PILLOW library, an image processing library, and sklearn.decomposition library, which is useful for its matrix optimization and decomposition functionality.



(a) Sample from non-uniform background test set

(b) Sample from plain white background test set

(c) Sample from a darker test set

(d) Sample from plain white background test set

Figure 3: Examples of the signed letter A T from two test sets with differing lighting and background

Image Enhancement: A combination of brightness, contrast, sharpness, and color enhancement was used on the images. For example, the contrast and brightness were changed such that fingers could be distinguished when the image was very dark.

Edge Enhancement: Edge enhancement is an image filtering techniques that makes edges more defined. This is achieved by the increase of contrast in a local region of the image that is detected as an edge. This has the effect of making the border of the hand and fingers, versus the background, much more clear and distinct. This can potentially help the neural network identify the hand and its boundaries.

Image Whitening: ZCA, or image whitening, is a technique that uses the singular value decomposition of a matrix. This algorithm decorrelates the data, and removes the redundant, or obvious, information out of the data. This allows for the neural network to look for more complex and sophisticated relationships, and to uncover the underlying structure of the patterns it is being trained on. The covariance matrix of the image is set to identity, and the mean to zero.

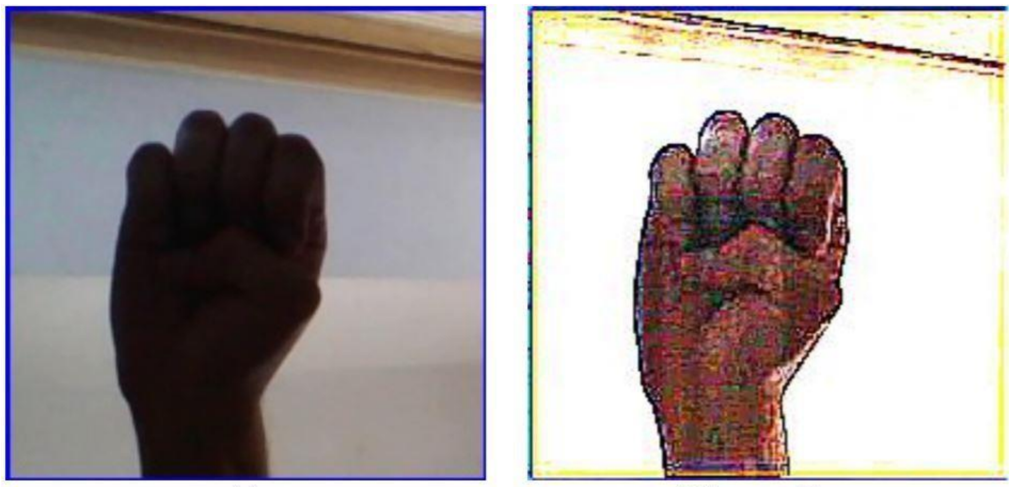

(a) (b) Preprocessed Image

Figure 4: Examples of image preprocessing.

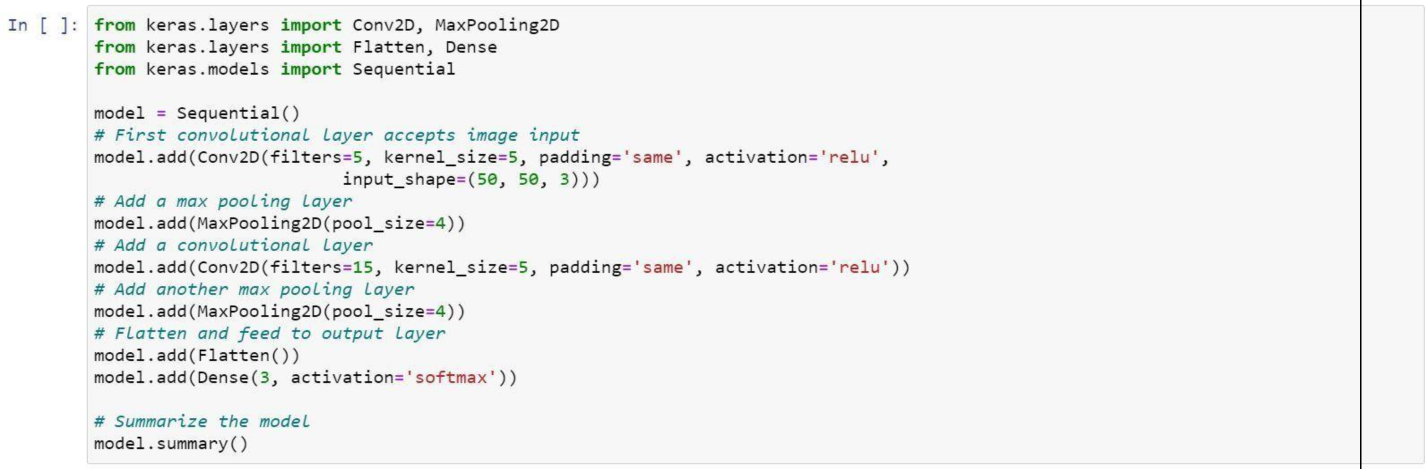
Data Processing

The load data.py script contains functions to load the Raw Image Data and save the image data as numpy arrays into file storage. The process data.py script will load the image data from data.npy and preprocess the image by resizing/rescaling the image, and applying filters and ZCA whitening to enhance features. During training the processed image data was split into training, validation, and testing data and written to storage. Training also involves a load dataset.py script that loads the relevant data split into a Dataset class. For use of the trained model in classifying gestures, an individual image is loaded and processed from the filesystem.

Training

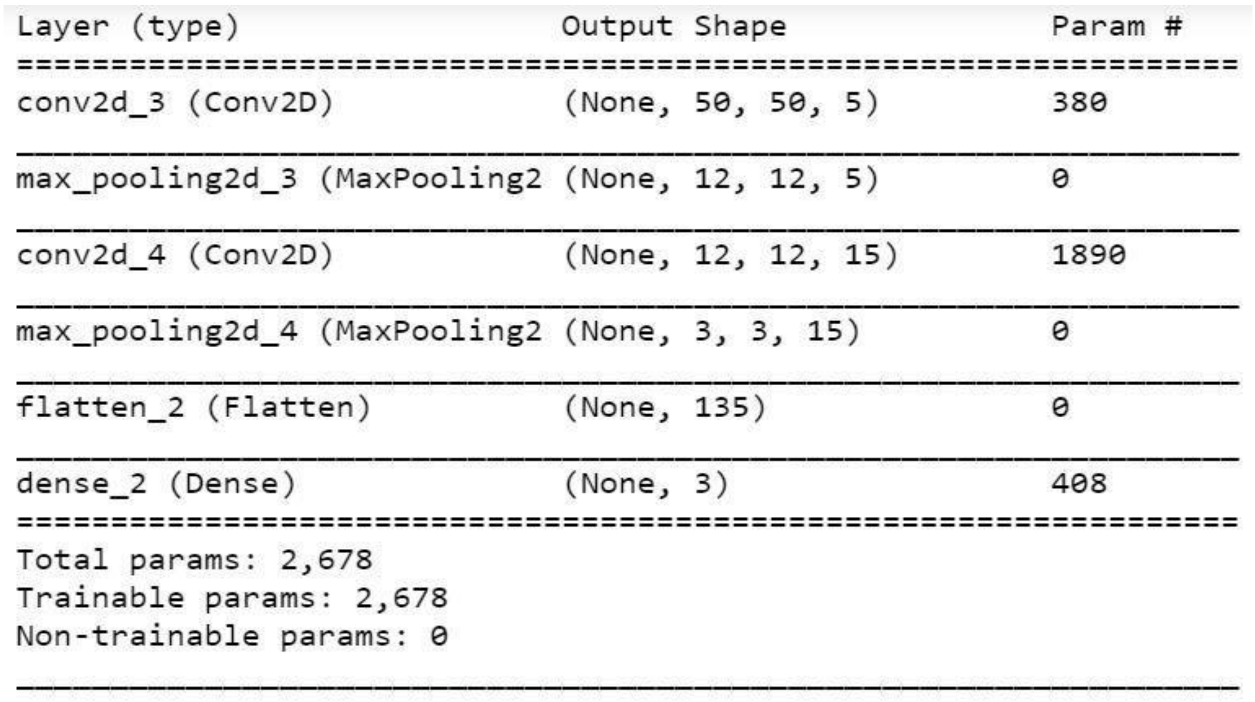
The training loop for the model is contained in train model.py. The model is trained with hyperparameters obtained from a config file that lists the learning rate, batch size, image filtering, and number of epochs. The configuration used to train the model is saved along with the model architecture for future evaluation and tweaking for improved results. Within the training loop, the training and validation datasets are loaded as Dataloaders and the model is trained using Adam Optimizer with Cross Entropy Loss. The model is evaluated every epoch on the validation set and the model with best validation accuracy is saved to storage for further evaluation and use. Upon finishing training, the training and validation error and loss is saved to the disk, along with a plot of error and loss over training.

Input

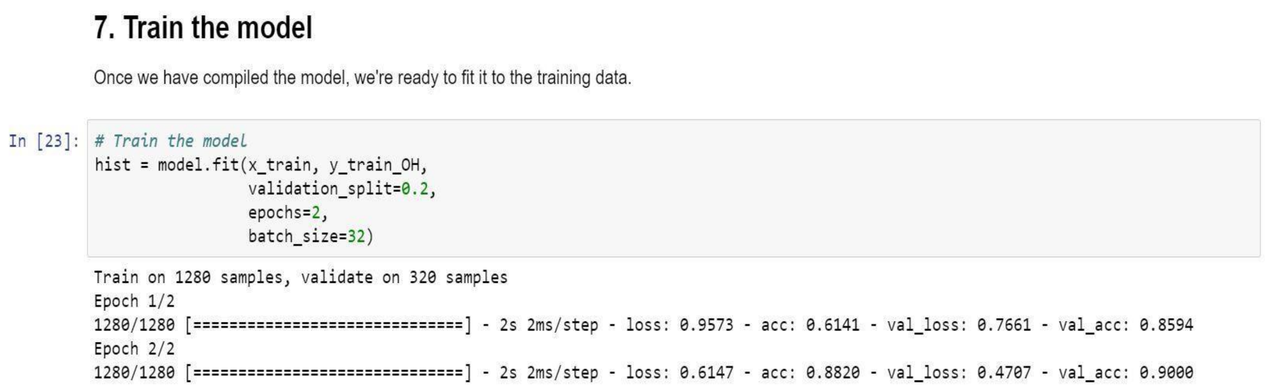


```
In [ ]: from keras.layers import Conv2D, MaxPooling2D
        from keras.layers import Flatten, Dense
        from keras.models import Sequential

        model = Sequential()
        # First convolutional layer accepts image input
        model.add(Conv2D(filters=5, kernel_size=5, padding='same', activation='relu',
                                input_shape=(50, 50, 3)))
        # Add a max pooling layer
        model.add(MaxPooling2D(pool_size=4))
        # Add a convolutional layer
        model.add(Conv2D(filters=15, kernel_size=5, padding='same', activation='relu'))
        # Add another max pooling layer
        model.add(MaxPooling2D(pool_size=4))
        # Flatten and feed to output layer
        model.add(Flatten())
        model.add(Dense(3, activation='softmax'))

        # Summarize the model
        model.summary()
```

Output



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 50, 50, 5)         380
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 5)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 15)        1890
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 15)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 135)               0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 408
=================================================================
Total params: 2,678
Trainable params: 2,678
Non-trainable params: 0
_________________________________________________________________
```

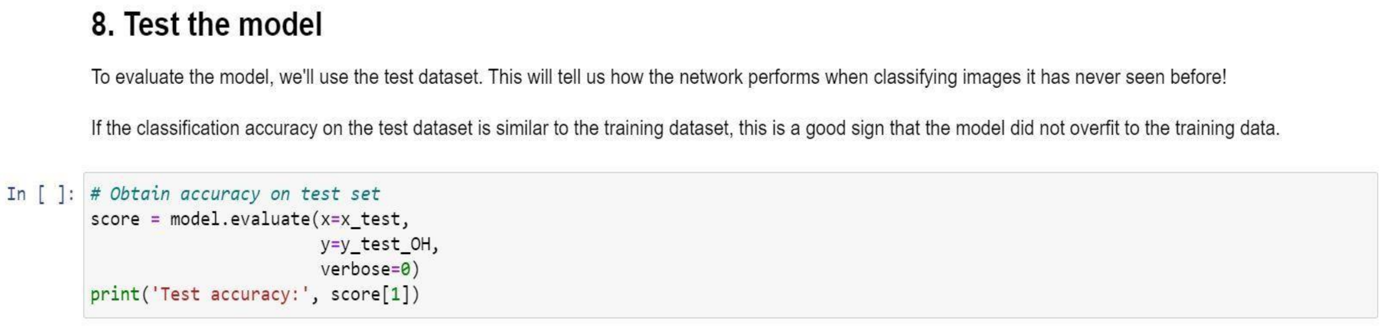


7. Train the model

Once we have compiled the model, we're ready to fit it to the training data.

```
In [23]: # Train the model
         hist = model.fit(x_train, y_train_OH,
                          validation_split=0.2,
                          epochs=2,
                          batch_size=32)
```

```
Train on 1280 samples, validate on 320 samples
Epoch 1/2
1280/1280 [==============================] - 2s 2ms/step - loss: 0.9573 - acc: 0.6141 - val_loss: 0.7661 - val_acc: 0.8594
Epoch 2/2
1280/1280 [==============================] - 2s 2ms/step - loss: 0.6147 - acc: 0.8820 - val_loss: 0.4707 - val_acc: 0.9000
```



8. Test the model

To evaluate the model, we'll use the test dataset. This will tell us how the network performs when classifying images it has never seen before!

If the classification accuracy on the test dataset is similar to the training dataset, this is a good sign that the model did not overfit to the training data.

```
In [ ]: # Obtain accuracy on test set
        score = model.evaluate(x=x_test,
                               y=y_test_OH,
                               verbose=0)
        print('Test accuracy:', score[1])
```

CONCLUSION AND FUTURE SCOPE

While this analysis sets a solid baseline for American Sign Language recognition, more work needs to be done to apply this concept in real time. This would require further work with the LeapMotion API to enable real-time generation of data, feeding through the model, and identification of the word and/or numbers. This would also require the model to be able to handle more than the 60 class labels it currently deals with.

In conclusion, we see this application having real potential in improving the lives of the hearing-impaired, and as such it would be a worthy goal to continue development.