

Final Report of Internship Program 2021

On

“American Sign Language”

26-02-2021

MEDTOUREASY

ACKNOWLEDGMENTS

The internship opportunity that I had with MedTourEasy was a great change for learning and understanding the intricacies of the subject of Data Visualizations in Data Analytics; 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 deepest gratitude and special thanks to the Training & Developement Team of MedTourEasy who gave me an opportunity to carry out my internship at their esteemed organization. Also, I express my thanks to 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 in spite of 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.

ABSTRACT

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. Therefore, to build 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 on classifying the Sign Language.

The goal of this project was to build a neural network able to classify which letter of the American Sign Language (ASL) alphabet is being signed, given an image of a signing hand. This project is a first step towards building a possible sign language translator, which can take communications in sign language and translate them into written and oral language. Such a translator would greatly lower the barrier for many deaf and mute individuals to be able to better communicate with others in day-to-day interactions.

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.

1.2 About the Project

Communication is very crucial to human beings, as it enables us to express ourselves. We communicate through speech, gestures, body language, reading, writing or through visual aids, speech being one of the most commonly used among them. However, unfortunately, for the speaking and hearing-impaired minority, there is a communication gap. Visual aids, or an interpreter, are used for communicating with them. However, these methods are rather cumbersome and expensive, and can't be used in an emergency. Sign Language chiefly uses manual communication to convey meaning. This involves simultaneously combining hand shapes, orientations and movement of the hands, arms or body to express the speaker's thoughts.

Sign Language consists of fingerspelling, which spells out words character by character, and word level association which involves hand gestures that convey the word meaning. Fingerspelling is a vital tool in sign language, as it enables the communication of names, addresses and other words that do not carry a meaning in word level association. In spite of this, fingerspelling is not widely used as it is challenging to understand and difficult to use. Moreover, there is no universal sign language and very few people know it, which makes it an inadequate alternative for communication.

This goal is further motivated by the isolation that is felt within the deaf community. Loneliness and depression exist in higher rates among the deaf population, especially when they are immersed in a hearing world. Large barriers that profoundly affect life quality stem from the communication disconnect

between the deaf and the hearing. Some examples are information deprivation, limitation of social connections, and difficulty integrating in society.

Most research implementations for this task have used depth maps generated by depth camera and high-resolution images. The objective of this project was to see if neural networks are able to classify signed ASL letters using simple images of hands taken with a personal device such as a laptop webcam. This is in alignment with the motivation as this would make a future implementation of a real time ASL-to-oral/written language translator practical in an everyday situation.

Trying to understand sign language from a first-vision perspective has the same limitations, some gestures will end up looking the same way. But this ambiguity can be solved by locating more cameras in different positions. In this way, what a camera can't see, can be perfectly observable by another camera.

Course Goals and Objectives:

1. Goal: To be able to initiate conversation and introduce themselves. Objective: Learn how to fingerspell their name. Objective: Know greetings and goodbyes.

2. Goal: To learn about others by asking personal questions.

Objective: Learn how to ask "yes/no" questions and the Facial Expressions that are required to ask those.

Objective: Learn how to ask “Wh-” questions and the Facial Expressions that are required to ask those. Objective: Analyzing the grammar behind the two types of questions and understanding the benefits of having different formats.

3. Goal: To communicate in American Sign Language.

Objective: Use scripted dialogue for students to first observe, then later, produce.

Objective: Initiate improvised dialogue for students to participate in.

Objective: Keeping a “voices off” environment in order to emerge students into the world of American Sign Language.

4. Goal: To learn about a new minority and what it is to be Deaf.

Objective: Compare and contrast different minorities with people who are Deaf and Hard of Hearing. Objective: Through videos, texts, experiences, and presentations; the many different ways to be Deaf or Hard of Hearing.

METHODOLOGY:

The gestures are performed by pressing the fingers either on the palm or other body parts such as forehead or in front of the chest. The words such as, „I“, „HELLO“, „YES“, „NO“, „THANK YOU“, „SORRY“ and „KNOW“ are the words for which signs can be performed and recognized using the proposed system.

☐ To convey the pronoun “I”, point the whole hand towards the chest while pressing the tips of index, middle and ring finger against the chest.

☐ To show the word “HELLO”, place the hand flat, touch the tip of the thumb below the small finger, press the tip of the index finger against forehead and then move the hand away from your forehead.

☐ To present the word “YES”, move the hand in up and down direction while pressing the middle, ring and little finger on the palm.

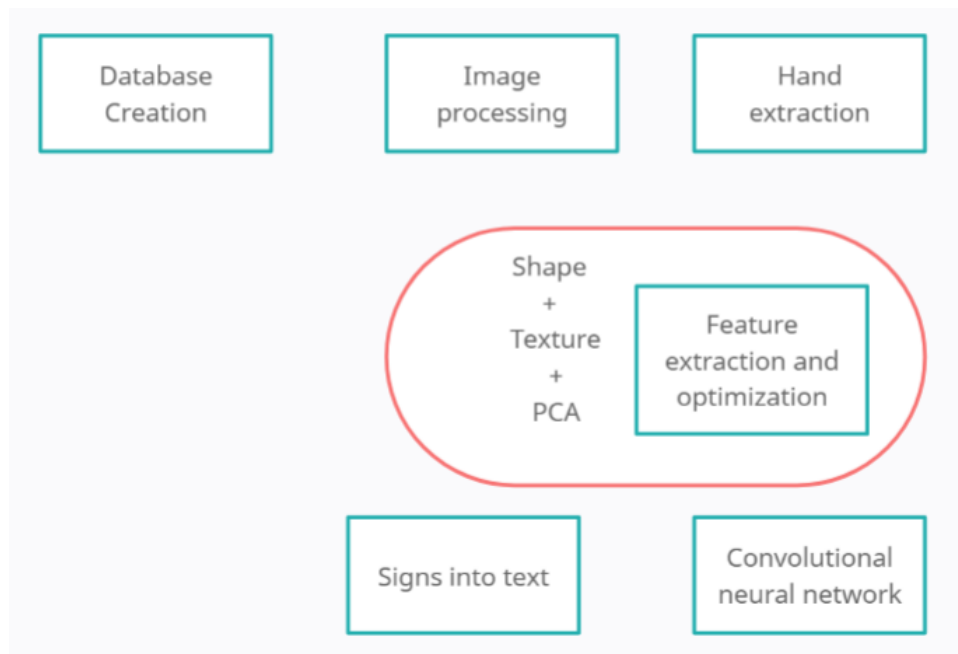
☐ To display the word “NO”, press the index and the middle finger against the thumb alongside bending the remaining two fingers.

☐ To indicate the word “SORRY”, make a fist and rub on the chest in clockwise direction.

☐ To communicate the word “KNOW”, press the index finger on the side of the forehead, bend the remaining fingers such that the thumb is pressed on the other three fingers (middle, ring & little)



Block diagram:



Language and Platform Used:

Language: Python

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was created in the late 1980s, and first released in 1991, by Guido van Rossum as a successor to the ABC programming language. Python 2.0, released in 2000, introduced new features, such as list comprehensions, and a garbage collection system with reference counting, and was discontinued with version 2.7 in 2020. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3. With Python 2's end-of-life (and pip having dropped support in 2021), only Python 3.6.x and later are supported, with older versions still supporting e.g. Windows 7 (and old installers not restricted to 64-bit Windows).

Python interpreters are supported for mainstream operating systems and available for a few more (and in the past supported many more). A global community of programmers develops and maintains CPython, a free and open-source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

IDE: Jupyter-notebook:

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library^[9] or "jupyter nbconvert" command line interface in a shell. To simplify visualization of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

II. IMPLEMENTATION:

Understanding the Data:

The second phase of our project will focus on dynamic signs (i.e., moving signs).

This dataset consists of 25 subjects each performing the same 60 ASL signs with both their left and right hands using a Leap Motion Controller (LMC). Therefore, this dataset has 60 different ASL signs (or class labels) that we are trying to accurately predict.

The LMC device records the position of the fingers, joints, palm, wrist, and arm every 0.04 seconds. In other words, the LMC acquires spatial coordinates of the skeleton joints of the hands and how these coordinates vary with time. Our second dataset is made up of these coordinate points.

Feature Engineering:

Realizing that we needed our data frames for each test subject to be comparable, we first transformed each data frame by taking the difference of each successive row of coordinates, giving the distance that each part of the hand (x, y, and z coordinates) traveled in between each measurement that the Leap Motion device recorded. This captured movement in intervals of approximately .04 seconds for the 54 parts of the hand identified during the motion capture. Each of these transformed data frames consisted of anywhere from 398–1203-time intervals, each with 162 columns of coordinate data.

After this differencing, we next sought to derive features that captured information about the movement of the hand during the given time interval. We decided to take the mean and standard deviation of each of these columns. While this may seem relatively simplistic, we found this a computationally cheap way to capture information.

Consider what the mean of each of these differenced columns represents. This is the average distance traveled by each part of the hand in each time interval. Likewise, taking the standard deviation of each of these columns represents the variation in this displacement. In other words, this serves as a proxy for velocity. (Surprisingly, adding the actual calculation for velocity at each point actually reduced accuracy!)

Learning/Modeling

Initially our team began model selection by looking at the subset of our data that consisted of numeric signs zero through ten, developing the features described above. What we realized is that this subset of signs could be easily distinguished by the fact that each of them only utilizes a single hand to complete the sign. Realizing this meant that the still left hand was only contributing noise to the data set, we removed all coordinates originating from the left hand and saw a significant gain in classification accuracy.

Now wanting to extend this to our full data set, we used both hands, again with the above features, and noted a significant decline in accuracy. In an attempt to identify where our model was unable to distinguish between different signs, we found that a better understanding of sign language would inform our model pipeline.

We first attempted to conditionally identify which signs utilized only one hand, with the intent of dividing our data set into two groups. Through both manually examining the signs and developing thresholds for our feature means, we split the data set into 22 “two-handed” signs and 38 “one-handed” signs by identifying data frames whose left-handed attributes appeared to be still, as measured by the sum of mean absolute deviation in left-handed coordinates. This distinction however, is often far from clear.

Model Selection

After creating the previously mentioned features, we experimented with several classifiers. Below are two boxplots, showing the performance of various models on our one-handed and two-handed signs. Ultimately, we decided that linear discriminant analysis (LDA) had superior performance. Below are our accuracy results, run for 100 test/training splits:

Screenshots of implementation:

1) American Sign Language (ASL):

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.

In this notebook, we will train a convolutional neural network to classify images of American Sign Language (ASL) letters. After loading, examining, and preprocessing the data, we will train the network and test its performance.

```
In [*]: # Import packages and set numpy random seed
import numpy as np
np.random.seed(5)
import tensorflow as tf
tf.set_random_seed(2)
from datasets import sign_language
import matplotlib.pyplot as plt
%matplotlib inline

# Load pre-shuffled training and test datasets
(x_train, y_train), (x_test, y_test) = sign_language.load_data()
```

2) Visualize the training data.

2. Visualize the training data

Now we'll begin by creating a list of string-valued labels containing the letters that appear in the dataset. Then, we visualize the first several images in the training data, along with their corresponding labels.

```
In [*]: # Store labels of dataset
labels = ['A', 'B', 'C']

# Print the first several training images, along with the labels
fig = plt.figure(figsize=(20,5))
for i in range(36):
    ax = fig.add_subplot(3, 12, i + 1, xticks=[], yticks=[])
    ax.imshow(np.squeeze(x_train[i]))
    ax.set_title("{} {}".format(labels[y_train[i]]))
plt.show()
```

3) Examine the dataset:

```
[~] ▶ MI  
  
# Number of A's in the training dataset  
num_A_train = sum(y_train==0)  
# Number of B's in the training dataset  
num_B_train = sum(y_train==1)  
# Number of C's in the training dataset  
num_C_train = sum(y_train==2)  
  
# Number of A's in the test dataset  
num_A_test = sum(y_test==0)  
# Number of B's in the test dataset  
num_B_test = sum(y_test==1)  
# Number of C's in the test dataset  
num_C_test = sum(y_test==2)  
  
# Print statistics about the dataset  
print("Training set:")  
print("\tA: {}, B: {}, C: {}".format(num_A_train, num_B_train, num_C_train))  
print("Test set:")  
print("\tA: {}, B: {}, C: {}".format(num_A_test, num_B_test, num_C_test))
```

4) One-hot encode the data:

Currently, our labels for each of the letters are encoded as categorical integers, where 'A', 'B' and 'C' are encoded as 0, 1, and 2, respectively.

```
y_test_OH = np_utils.to_categorical(y_test, 3)
```

5) Define the model:


Now it's time to define a convolutional neural network to classify the data.

This network accepts an image of an American Sign Language letter as input. The output layer returns the network's predicted probabilities that the image belongs in each category.

```
[~] ▶ M4  
  
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  
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()
```


6) Compile the model:

After we have defined a neural network in Keras, the next step is to compile it!

```
[ - ] ▶  M4  
# Compile the model  
model.compile(optimizer='rmsprop',  
              loss='categorical_crossentropy',  
              metrics=['accuracy'] )
```

7) Train the model:

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

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!

7. Train the model

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

```
In [0]: # Train the model
hist = model.fit(x_train, y_train_OH, validation_split=0.2, epochs=2, batch_size = 32)
```

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

8) Visualize mistakes:

Hooray! Our network gets very high accuracy on the test set!

The final step is to take a look at the images that were incorrectly classified by the model. Do any of the mislabeled images look relatively difficult to classify, even to the human eye?

Sometimes, it's possible to review the images to discover special characteristics that are confusing to the model. However, it is also often the case that it's hard to interpret what the model had in mind!

9. Visualize mistakes

Hooray! Our network gets very high accuracy on the test set!

The final step is to take a look at the images that were incorrectly classified by the model. Do any of the mislabeled images look relatively difficult to classify, even to the human eye?

Sometimes, it's possible to review the images to discover special characteristics that are confusing to the model. However, it is also often the case that it's hard to interpret what the model had in mind!

```
In [0]: # Get predicted probabilities for test dataset
y_probs = model.predict(x_test)

# Get predicted labels for test dataset
y_preds = np.argmax(y_probs, axis=-1)

# Indices corresponding to test images which were mislabeled
bad_test_idxs = np.where(y_preds!=y_test)[0]

# Print mislabeled examples
fig = plt.figure(figsize=(25,4))
for i, idx in enumerate(bad_test_idxs):
    ax = fig.add_subplot(2, np.ceil(len(bad_test_idxs)/2), i + 1, xticks=[], yticks=[])
    ax.imshow(np.squeeze(x_test[idx]))
    ax.set_title("{} (pred: {})".format(labels[y_test[idx]], labels[y_preds[idx]]))
```

DATASET

To develop the first prototype of this system it was used a dataset of 24 static signs from the Panamanian Manual Alphabet.



Data collection and preprocessing

To collect the dataset, it was asked to four users to wear the vision system and perform every gesture for 10 seconds while both cameras were recording in a 640x480 pixel resolution.

It was requested to the users to perform this process in three different scenarios: indoors, outdoors, and in a green background scenario. For the indoors and outdoors scenarios the users were requested to move around while performing the gestures in order to obtain images with different backgrounds, light sources,

and positions. The green background scenario was intended for a data augmentation process, we'll describe later.

After obtaining the videos, the frames were extracted and reduced to a 125x125 pixel resolution.

Data augmentation

Since the preprocessing before going to the convolutional neural networks was simplified to just rescaling, the background will always get passed to the model. In this case, the model needs to be able to recognize a sign despite the different backgrounds it can have.

To improve the generalization capability of the model it was artificially added more images with different backgrounds replacing the green backgrounds. This way it is obtained more data without investing too much time.

Evaluation

The models were evaluated in a test set with data corresponding to a normal use of the system in indoors, in other words, in the background it appears a person acting as the observer, similar to the input image in the figure above (Convolutional neural networks architecture).

Although the model learned to classify some signs, such as Q, R, H; in general, the results are kind of discouraging. It seems that the generalization capability of the

models wasn't too good. However, the model was also tested with real-time data showing the potential of the system.

Conclusions

Sign language recognition is a hard problem if we consider all the possible combinations of gestures that a system of this kind needs to understand and translate. That being said, probably the best way to solve this problem is to divide it into simpler problems, and the system presented here would correspond to a possible solution to one of them.

The system didn't perform too well but it was demonstrated that it can be built a first-person sign language translation system using only cameras and convolutional neural networks.

It was observed that the model tends to confuse several signs with each other, such as U and W. But thinking a bit about it, maybe it doesn't need to have a perfect performance since using an orthography corrector or a word predictor would increase the translation accuracy.

Future Steps

- Use Dynamic Loading for Dataset: Our original dataset was quite large and is impossible to use without a server with a lot of RAM and disk space. A possible solution is to split the file names into training, validation, and test sets and dynamically loading images in the Dataset class. Using such a loading technique would allow us to train the model on more samples in the dataset.