Project Proposal

Participants:

Yoon Chi, Anjali Abraham, Isaac Nkrumah

Objective:

Our project's objective is to utilize computer vision libraries (i.e., PyTorch) and Python libraries to train a model to interpret and translate gestured American sign language (ASL) into written English. This project is relevant and important as we are still in the process of bridging the communication gap that currently exists between the deaf and non-deaf communities. Our goal is to promote understanding and self-awareness of ASL, as well as to create a simple tool that can translate basic ASL alphabet and sentences.

Related Work:

Sign language detector to detect live sign gestures as primary regions of interest, and to ignore foreground, was successfully created with just OpenCV library and Keras modules of python ("*Sign Language Recognition Using Python and OpenCV", Data-Flair, 2021*). Limitations in sign language recognition include dependency on distance (as distance of sign from live web can increase or decrease), background interference (accuracy decreases as background becomes noisy), and environment changes (interpretation of a sign against environment 1 vs. environment 2 are different) (*"Using Computer Vision to Help Deaf and Hard of Hearing Communities", Joseph Nelson, 2020*).

Technical Overview:

We will be using the supervised learning paradigm based on the training samples. The goal of this paradigm is to create a system that maps and searches for patterns with the input samples which would be the images we collected, and then correlates with the desired outputs. After training, the system will predict new outputs for new unseen inputs, and determine which label the new inputs will be classified based on prior training. We are not limiting this project to only the supervised learning paradigm, as we are learning more models that we can implement in this project.

We will mimic Data-Flair's project for the first half of our project, and then focus on improving the accuracy of the sign language detection by targeting the limitations that Joseph Nelson faced. For the first half of our project, we will begin with dividing the project into three parts: creating the dataset, training the model, and then predicting the data. Our dataset will be created using grayscale images of signs or hand gestures, each taken from the top, bottom, left and right sides. Grayscale images give a higher accuracy in classification. To get these images, we will be using OpenCV and creating an ROI—the part of the frame that detects the hand gestures. Then, we would capture the gestures, and differentiate it from the background and subtract from the frames. Next, we would define and detect the outermost contours of the hand and return it. Then save the image of the ROI in the train and test set.

The second part of the project is to train the dataset in a CNN model. First, we would load the train and test samples, and plot the images. We would then design the CNN model by using the following architectures; Conv2D, Dense, Flatten and Dropout. Then, we would compile and fit the model. Lastly, we will be predicting the gestures by creating a bounding box for detecting the ROI and creating new datasets. Then, we will test the datasets and compare them with the models, and determine the gestures.

As for the libraries and software, we will use Pytorch to build an LSTM neural network, Onnx to export the neural network, OpenCV to process the labeled images, and Tensorflow for object recognition. We will use Google Collab as our primary IDE; with Collab, we have free access to computing GPU resources. If we have time and succeed in improving the accuracy of sign detection, we can focus on interpreting the data from live-feed. We would need a videocam. If we opt to train our model real-time, we won't need to keep all training material in memory. We could therefore save memory for deep learning computations. Apache-Kafka may be useful in this case.

Expected use of deep neural networks, in terms of input and output: the input will primarily consist of static images of different signs that will be stored and fed to the model as single-channel images (this can come from already available datasets - dataset 1, and dataset 2). Additionally, we can take videos of ourselves, or existing videos of people gesturing sign language, splice the videos into separate static images, and then feed the model the static images in order. Expected outputs include ASL experts confirming the model's translations from ASL to English.

Citations:

- Bheda, Vivek, and Dianna Radpour. “Using Deep Convolutional Networks for Gesture Recognition in American Sign Language.” *ArXiv.org*, 20 Nov. 2017, https://arxiv.org/abs/1710.06836.
- *View of American Sign Language Recognition Using CNN*, http://www.journals.resaim.com/ijresm/article/view/92/80.
- “Sign Language Recognition Using Python and OpenCV.” *DataFlair*, 25 Aug. 2021, https://data-flair.training/blogs/sign-language-recognition-python-ml-opencv/.
- Nelson, Joseph. “Using Computer Vision to Help Deaf and Hard of Hearing Communities.” *Roboflow Blog*, Roboflow Blog, 28 Oct. 2020, https://blog.roboflow.com/computer-vision-american-sign-language/.