LITERATURE SURVEY

Date	24 September 2022
Team ID	PNT2022TMID39415
Project Name	Real-Time communication System Powered By AI For specially abled

INTRODUCTION:

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. This goal is further motivated by the isolation that is felt within the deaf community. Loneliness and depression exists in higher rates among the deaf population, especially when they are immersed in a hearing world [1]. 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 [2]. 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.

Description of Overall Software Structure:

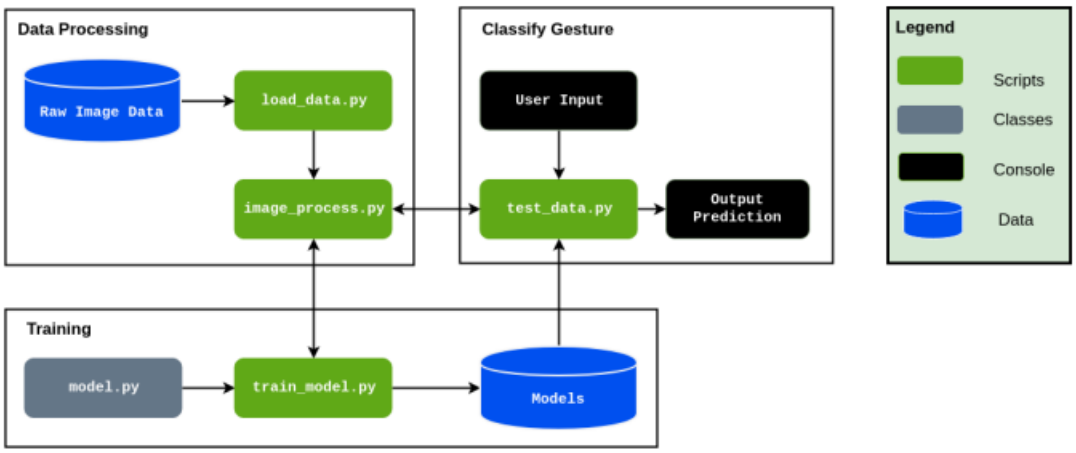


Figure 1: Block Diagram of Software

As shown in Figure 1, the project will be structured into 3 distinct functional blocks, Data Processing, Training, Classify Gesture. The block diagram is simplified in detail to abstract some of the minutiae:
• 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.

• Classify Gesture: After a model has been trained, it can be used to classify a new ASL gesture that is available as a file on the filesystem. The user inputs the filepath of the gesture image and the test data.py script will pass the filepath to process data.py to load and preprocess the file the same way as the model has been trained.

Sources of Data

3.1 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 labelled for use.

(a) ASL letter A (b) ASL letter E (c) ASL letter H (d) ASL letter Y

3.2 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:

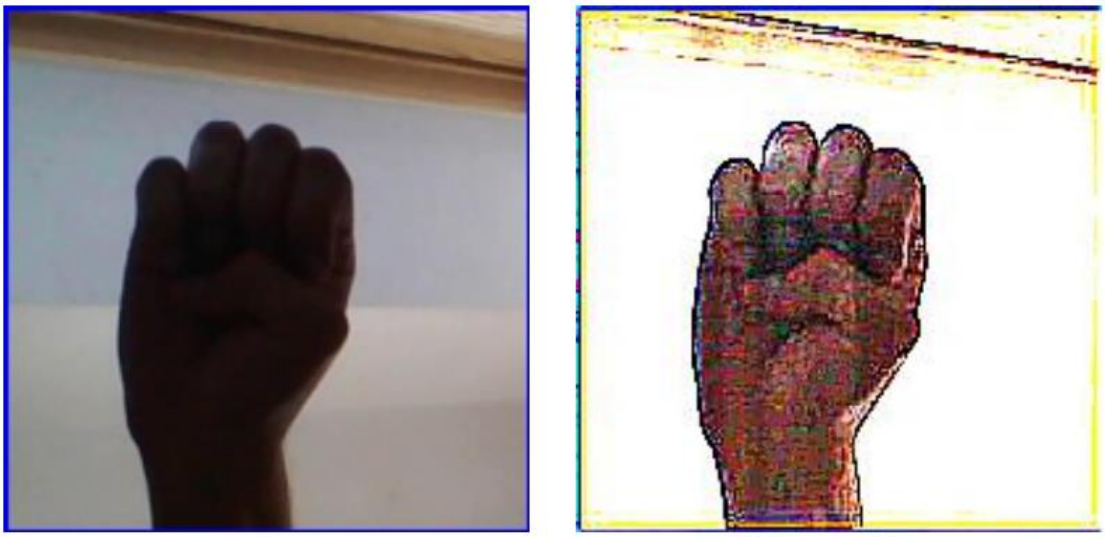
Edge enhancement is an image filtering techniques that makes edges more defined. This is achieved by the increase of contrast in a local region of the image that is detected as an edge. This has the effect of making the border of the hand and fingers, versus the background, much more clear and distinct.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

Machine Learning Model 4.1 Overall Structure The model used in this classification task is a fairly basic implementation of a Convolutional Neural Network (CNN). As the project requires classification of images, a CNN is the go-to architecture. The basis for our model design came from Using Deep Convolutional Networks for Gesture Recognition in AmericanSign Language paper that accomplished a similar ASL Gesture Classification task [4]. This model consisted of convolutional blocks containing two 2D Convolutional Layers with ReLU activation, followed by Max Pooling and Dropout layers. These convolutional blocks are repeated 3 times and followed by Fully Connected layers that eventually classify into the required categories. The kernel sizes are maintained at 3 X 3 throughout the model. Our originally proposed model is identical to the one from the aforementioned paper, this model is shown in Figure 5. We omitted the dropout layers on the fully connected layers at first

to allow for faster training and to establish a baseline without



(a) (b) Preprocessed Image

Figure 4: Examples of image preprocessing.

Machine Learning Model

4.1 Overall Structure

The model used in this classification task is a fairly basic implementation of a Convolutional Neural Network (CNN). As the project requires classification of images, a CNN is the go-to architecture. The basis for our model design came from Using Deep Convolutional Networks for Gesture Recognition in AmericanSign Language paper that accomplished a similar ASL Gesture Classification task [4]. This model consisted of convolutional blocks containing two 2D Convolutional Layers with ReLU activation, followed by Max Pooling and Dropout layers. These convolutional blocks are repeated 3 times and followed by Fully Connected layers that eventually classify into the required categories. The kernel sizes are maintained at 3 X 3 throughout the model. Our originally proposed model is identical to the one from the aforementioned paper, this model is shown in Figure 5. We omitted the dropout layers on the fully connected layers at first to allow for faster training and to establish a baseline without dropout.

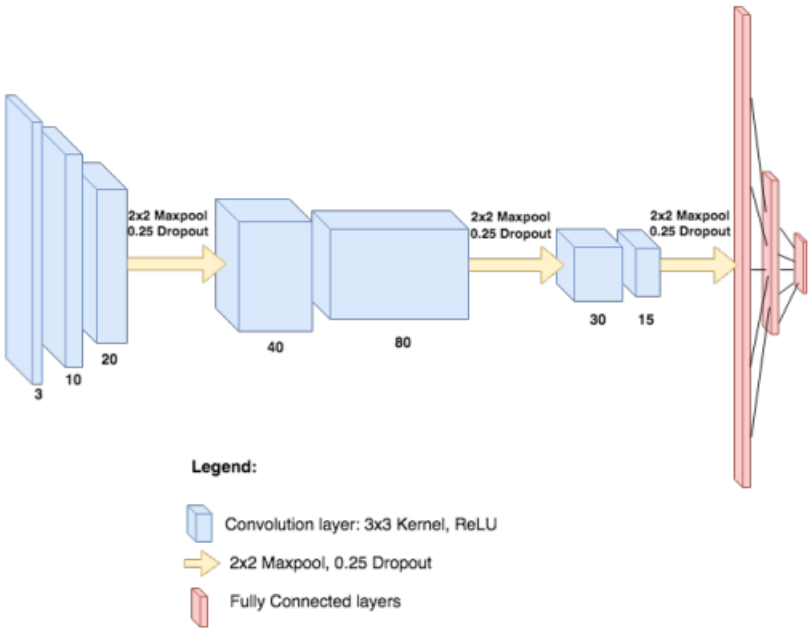


Figure 5:

Model Architecture as implemented in Using Deep Convolutional Networks for Gesture Recognition in AmericanSign Language[4]

We also decided to design a separate model to compare with the model in the paper. This model was designed to be trained faster and to establish a baseline for problem complexity. This smaller model was built with only one “block” of convolutional layers consisting of two convolutional layers with variable kernel sizes progressing from 5 X 5 to 10 X 10, ReLU activation, and the usual Max Pooling and Dropout. This fed into three fully connected layers which output into the 29 classes of letters. The variation of the kernel sizes was motivated by our dataset including the background, whereas the paper preprocessed their data to remove the background. The design followed the thinking that the first layer with smaller kernel would capture smaller features such as hand outline, finger edges and shadows. The larger kernel hopefully captures combinations of the smaller features like finger crossing, angles, hand location, etc. This model architecture is shown in Figure 6.

CONCLUSION

• 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.