Machine Vision:
A Tool for Automated Sign Language Interpretation

Jason Watson

CS6318 – Artificial Intelligence

December 5, 2021

Application Description

Deaf people across the world have a limited number of options for telecommunications including: Video Relay Service (VRS), Video Remote Interpreting (VRI), and the text based telecommunications relay service (TRS).[1] All these solutions require a person to serve as the interpreter between the deaf person and their contact.

The goal of this application is to interpret the first four letters in the ASL finger spelling alphabet using artificial intelligence and neural networks that can run efficiently on mobile devices. The program should output the interpretation to the screen and an output file. The first four letters were chosen arbitrarily and would serve as a proof of concept if successful. If this solution can work on mobile devices, it would increase the accessibility to a larger audience of potential users and use cases.

Restrictions

1. Only one letter should be returned by the program for any given sign.
2. Only output predictions if they exceed or meet the minimum threshold of 90%.
3. Only output a space if the confidence of the current sign drops below or to ¼ of the minimum allowed threshold at least once after the previous letter prediction.
4. Do not duplicate the previous output.

[1] National Deaf Center, *Telecommunications: VRS, VRI, and TRS*, accessed 12/2020

Actions

1. Interpret sign.
2. Ignore input.

Goal

Interpret ASL fingerspelling of A, B, C, and D individually through a live video feed and output those results to the screen and a log file.

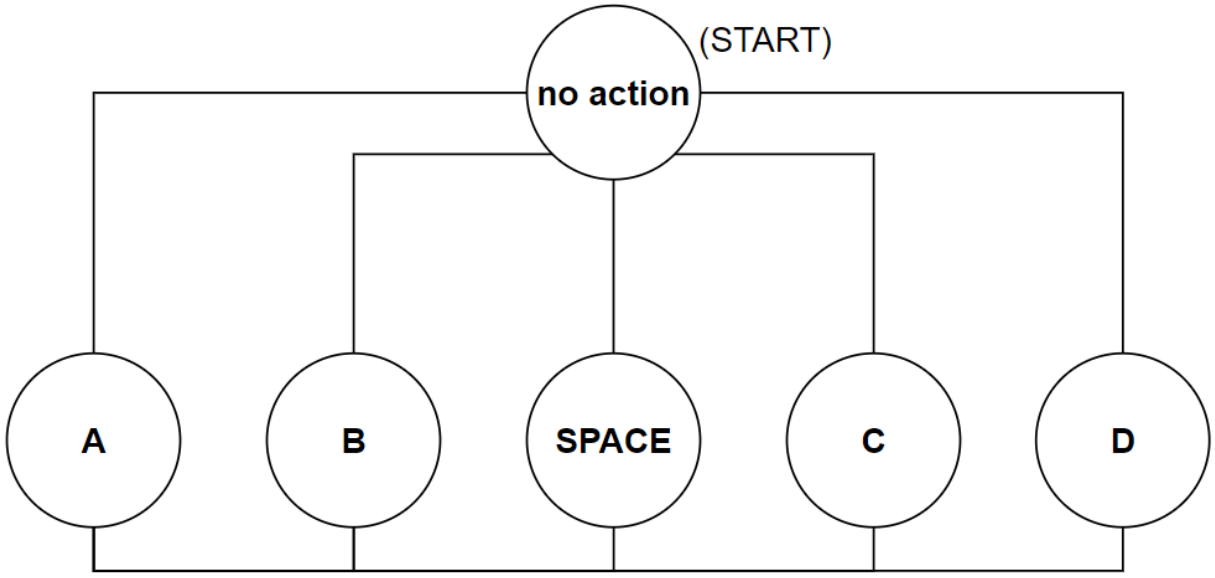
Metrics

A trained neural network will provide an array of prediction/confidence percentages and another array of the corresponding labels. The first result in the prediction array is the most probable.

Knowledge-base

1. If [LastDetection is not X] then [X is a new detection]
2. If [confidence([X is a new detection]) is >= 90%] then [X is a letter]
3. If [X is a letter] then [X is now the LastDetection]
4. If [confidence([X is a new detection]) <= 22.50%] then [X is a space]
5. If [X is a space] then [X is now the Last Detection]

State-Space Graph



Forward Chaining

KB

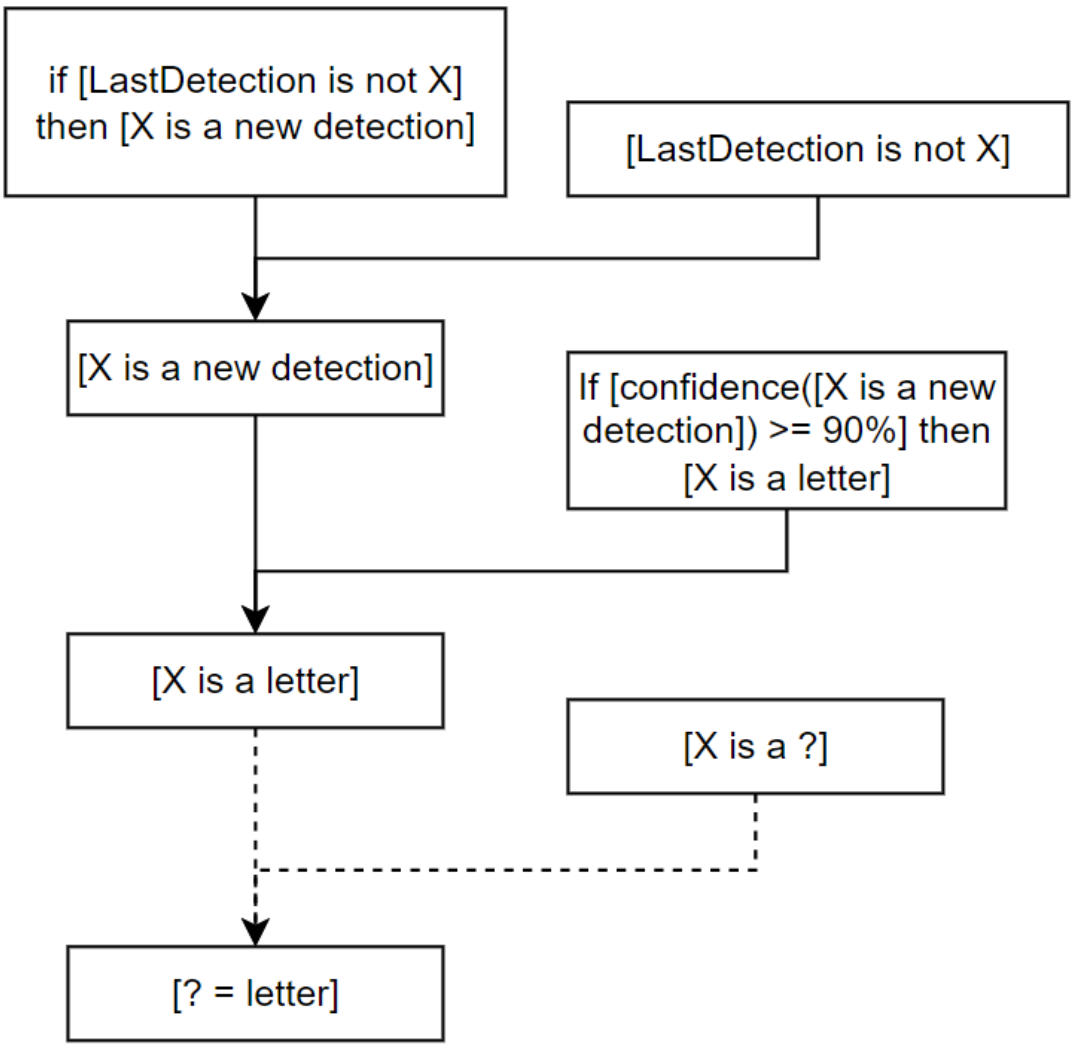
1. If [LastOutput is not X] then [X is a new detection]
2. If [confidence([X is a new detection]) is >= 90%] then [X is a letter]
3. If [X is a letter] then [X is now the LastOutput]
4. If [confidence([X is a new detection]) <= 22.50%] then [X is a space]
5. If [X is a space] then [X is now the LastOutput]

Input: [LastOutput is not X]

[X is a new detection]

[X is a letter]

Goal: [X is a ?]



Sample Usage

The application is presented with an image from a live video feed. What ASL finger spelling sign does it contain, if any?

The agent takes in the image and uses the trained neural network to identify potential finger spellings and their corresponding probabilities. The neural network returns three arrays, one of predictions, one of the corresponding probabilities, and one of the corresponding detection boxes.

Predictions:

[1 3 3 2 2 3 4 3 3 4 1 1 3 1 3 4 2 3 1 4 2 1 3 1 4 2 1 3 1 3 3 1 2 4 2 1 1
2 4 2 3 4 4 4 3 3 1 3 4 3 4 1 3 1 3 4 3 1 3 4 3 3 2 4 4 2 3 3 4 3 1 3 1 3
3 4 3 4 4 4 1 1 4 4 3 3 3 2 4 2 3 2 4 4 2 4 4 1 4]

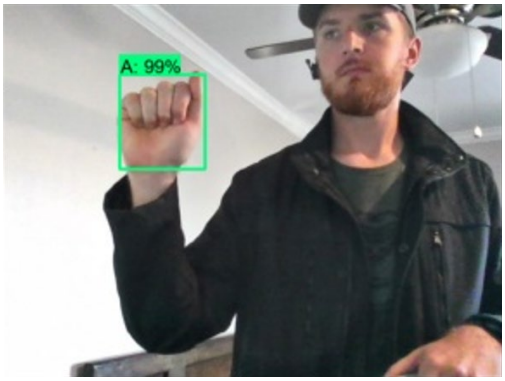
Probabilities:

[0.98501694 0.10014024 0.07919845 0.05478474 0.04315928 0.03475374
0.03358316 0.03135094 0.03080592 0.02945182 0.02919823 0.02641395
0.02463454 0.02428254 0.02387434 0.0199486 0.01990354 0.01947239
0.01930988 0.01906672 0.01888567 0.01872364 0.01850909 0.01804385
0.01766536 0.01757073 0.01727369 0.01722413 0.01626557 0.01592341
0.01587796 0.01571733 0.01480708 0.01444119 0.01421878 0.01392153
0.01380131 0.01361305 0.01303092 0.01207083 0.01203349 0.01186392
0.01155257 0.0114938 0.01136348 0.01127729 0.01119813 0.01110196
0.01106381 0.01087639 0.0107778 0.01077235 0.01050878 0.01020283
0.00997263 0.00987086 0.00978836 0.00977883 0.00977138 0.00958475
0.00955027 0.00949314 0.00940508 0.00937051 0.00930139 0.00925413
0.00920695 0.00918582 0.00917229 0.00903052 0.00899419 0.00888929
0.00883418 0.00879192 0.00876722 0.00873414 0.00868717 0.00863054
0.00859961 0.00854322 0.0084298 0.00829917 0.00827509 0.00820041
0.00818264 0.00813869 0.00808966 0.00800303 0.0079011 0.00788775
0.00787678 0.00787026 0.0076932 0.00756121 0.00752851 0.00751078
0.00745732 0.00744697 0.0074406 0.00740317]

Detection boxes (truncated):

[[0.19857548 0.23672861 0.44751704 0.40904796],…]

The first and most probable prediction is considered. The neural network indicates that there is a 99% probability that fingerspelling for A is in the picture. The application draws a box according to the detection box location points indicated in the detection box array, labels the box according to the prediction array, and appends the corresponding probability. This new version of the image is displayed in the live feed video so the user can see real time detections. The application then writes the prediction to the output.txt file. The display of the detection box is not part of the AI actions, rather the perception of its environment. The AI uses predicate logic and forward chaining to determine if the letter should be output to the output.txt file.



Conclusion and Discussion

This project was challenging and rewarding to research. I originally planned to create program that would interpret all 26 letters of the ASL fingerspelling alphabet. I couldn't find a good dataset that contained enough information on all the letters. Therefore, I opted to reduce the scope to the first four letters and create my own dataset. In total, I collected approximately 170 photos of the four letters. Each of the photos was annotated by hand to identify the precise location of the sign. I used a free program available on github called LabelImg to annotate each image.[2] After creating my dataset, I augmented the data by applying filters to each image. I applied three different filters to create an additional 3 photos for every image. Data augmentation increased my dataset to approximately 680 unique images and improved the networks mean average precision.

[2] Tzutalin, LabelImg, https://github.com/tzutalin/labelImg

I used the tensorflow object_detection library and one of their pretrained models to train my neural network. I opted to start with their ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8 pretrained model. This model is optimized for mobile devices and should perform better on devices that are computationally limited.[3] I trained the neural network for 8,000 epochs on my dataset. In the end, my model had a mean average precision of 80%. A larger dataset would improve the mean average precision. With the model trained, I was able to take images from a live video feed and make predictions. I could then use the object_detection library to draw boxes around the detected object and display them to the screen. The AI for this project used the neural network as a way to perceive it's environment. The main goal of the AI was to perceive the environment and based on those perceptions, take an action. In this case, we chose to either output the perception to a text file or ignore the perception. Even though this dataset is extremely small for machine vision standards, we were able to predict the letters A, B, C, and D with encouraging accuracy. This project has shown that given a sufficiently large dataset, a program could interpret static sign language gestures. Further research should investigate the ability to detect complex signs that involve movement. Most complex sign language gestures involve movement and the ability to accurately interpret these signs would open the door for a suite of tools that could improve the daily lives of millions of people.

[3] TensorFlow, Object Detection, accessed 12/2020

Works Cited

National Deaf Center, *Telecommunications: VRS, VRI, and TRS*, accessed 12/2020, https://www.nationaldeafcenter.org/sites/default/files/Telecommunications_%20VRS,%20VRI,%20and%20TRS.pdf

Tzutalin, LabelImg, https://github.com/tzutalin/labelImg

TensorFlow, Object Detection, accessed 12/2020, https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

Program to run application (should also be included in BB submission):

Run.py

```
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
import cv2
import numpy as np
from datetime import datetime

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections

labels = [{'name':'A', 'id':1}, {'name':'B', 'id':2}, {'name':'C', 'id':3},
{'name':'D', 'id':4}]

# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file('pipeline.config')
detection_model = model_builder.build(model_config=configs['model'],
is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore('ckpt-9').expect_partial()
category_index =
label_map_util.create_category_index_from_labelmap('label_map.pbtxt')

cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
lastDetection = 0
minScore = 90

f = open("output.txt", "a")
newLog = '\n\n' + datetime.now().strftime("%m/%d/%Y, %H:%M:%S") + '\n'
f.write(newLog)
while cap.isOpened():
    ret, frame = cap.read()
    image_np = np.array(frame)

    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0),
dtype=tf.float32)
    detections = detect_fn(input_tensor)

    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
```

```
    detections['num_detections'] = num_detections

    # detection_classes should be ints.
    detections['detection_classes'] =
detections['detection_classes'].astype(np.int64)

    label_id_offset = 1
    image_np_with_detections = image_np.copy()

    currentScore = round(100 * detections['detection_scores'][0])
    currentDetection = detections['detection_classes'][0]+label_id_offset
    if(currentScore > minScore and lastDetection != currentDetection ):
        for item in labels:
            if(item['id'] == currentDetection):
                print("{}".format(item['name']),end='')
                f.write(item['name'])
                lastDetection = currentDetection
    elif(currentScore < minScore/4 and lastDetection != 0):
        lastDetection = 0
        print(" ",end='')
        f.write(' ')

    viz_utils.visualize_boxes_and_labels_on_image_array(
                image_np_with_detections,
                detections['detection_boxes'],
                detections['detection_classes']+label_id_offset,
                detections['detection_scores'],
                category_index,
                use_normalized_coordinates=True,
                max_boxes_to_draw=5,
                min_score_thresh= minScore/100,
                agnostic_mode=False)

    cv2.imshow('object detection',  cv2.resize(image_np_with_detections,
(800, 600)))

    if cv2.waitKey(10) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()
        break
f.close()
```