Collin Li 1/1/2022

Maker Portfolio Documentation

MAKER VIDEO: https://youtu.be/RLcMY8iDpvs

Individual - (Recent) Academic Paper

First-author (in review), IRIS – Intelligent Rapid Interactive Segmentation for measuring liver cyst volumes in autosomal dominant polycystic kidney disease. Tomography.

CollinLi_tomography-journal_IRIScv.pdf← This is a link (links like this will be in blue below)

Abstract

Purpose: To develop and integrate interactive features with automatic methods for accurate liver cyst segmentation in patients with autosomal dominant polycystic kidney and liver disease (ADPKD).

Methods: SmartClick and antiSmartClick were developed using iterative region growth guided by spatial and intensity connections and were integrated with the automated level set (LS) segmentation and graphical user interface, forming an Intelligent Rapid Interactive Segmentation (IRIS) tool. IRIS and LS segmentations of liver cysts on TT2 weighted images of patients with ADPKD (n=17) were compared with manual segmentation as ground truth (GT).

Results: Compared to manual GT, IRIS reduced the segmentation time by more than 10-fold. Compared to automated LS, IRIS reduced the mean liver cyst volume error from 42.22% to 13.44% ($p<0.001$). IRIS segmentation agreed well with manual GT (79% dice score and 99% intra-class correlation coefficient).

Conclusion: IRIS is feasible for fast, accurate liver cyst segmentation in patients with ADPKD.

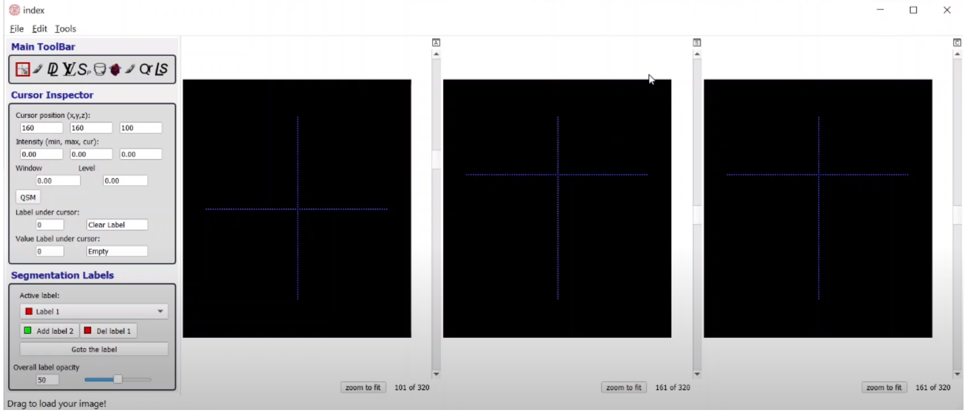
Continuation

As I am still working on improving the workflow of segmentation, I have discovered that using a rough level set segmentation tool in addition to SmartClick can reduce segmentation times to under 5 minutes.

Youtube Video Demonstrating the liver cyst segmentation of a difficult MRI image in less than 3 minutes

Youtube Video Demonstrating features and workflow of the application

Demo images of application (from youtube video):



Cursor position (x,y,z): 160 160 100

Intensity (min, max, cur): 0.00 0.00 0.00

Window 0.00 Level 0.00

Label under cursor: 0 Clear Label

Value Label under cursor: 0 Empty

Active label: Label 1

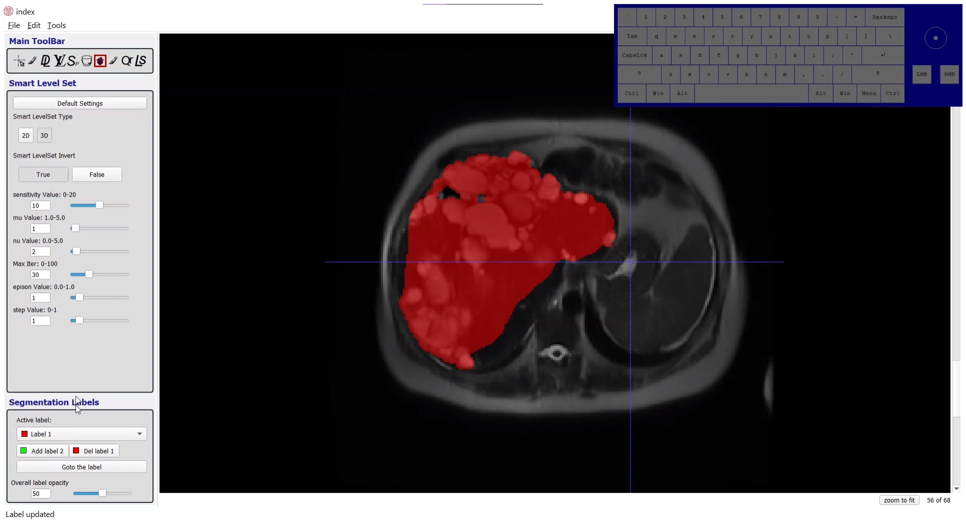
Overall label opacity 50

101 of 320

161 of 320

161 of 320

Drag to load your image!



Smart LevelSet Type

Smart LevelSet Invert

sensitivity Value: 0-20 10

mu Value: 1.0-5.0 1

nu Value: 0.0-5.0 2

Max Iter: 0-100 30

epison Value: 0.0-1.0 1

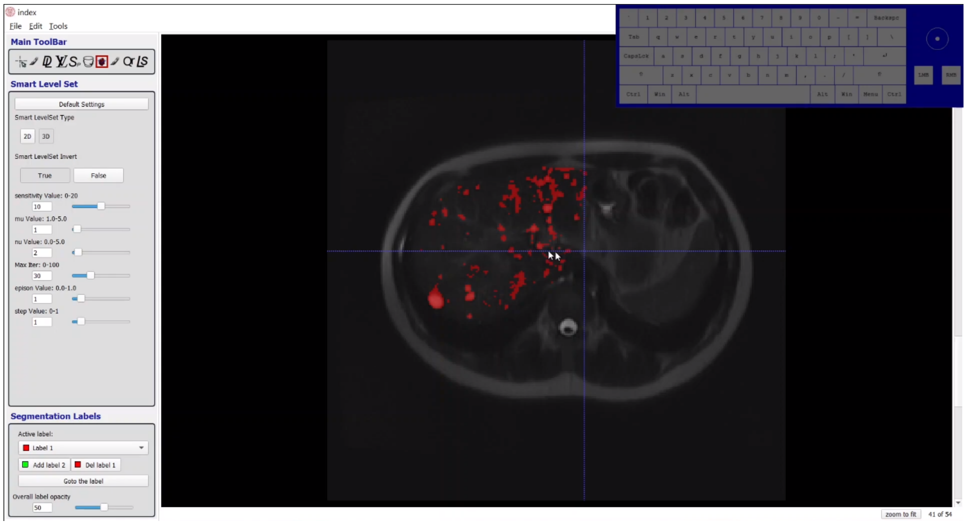
step Value: 0-1 1

Active label: Label 1

Overall label opacity 50

56 of 68

Label updated



Smart LevelSet Type

Smart LevelSet Invert

sensitivity Value: 0-20 10

mu Value: 1.0-5.0 1

nu Value: 0.0-5.0 2

Max Iter: 0-100 30

epison Value: 0.0-1.0 1

step Value: 0-1 1

Active label: Label 1

Overall label opacity 50

41 of 54

Individual - Liver Cyst Segmentation @ Weill Cornell MRI Lab

Since Summer between 11th and 12th Grade

CollinLi_ResearchRoundtables_ppt.pdf

QT-SmartClick: A Novel Automated Segmentation Tool for Medical Images with Interactive Optimization Capabilities Designed for Liver Lesions

Under the mentorship of Professor Wang and a radiology graduate student, I designed algorithms to automate the tedious process of segmentation.

Description

Liver Cancer is a prevalent disease and one of the most common cancers. Magnetic Resonance Imaging (MRI) is a common method used to detect anomalies and liver tumors. A crucial process of analyzing these abdominal scans is detecting and labeling the liver and liver tumors, known as segmentation. This is performed for volumetric analysis and preparation for surgeries. Currently, segmentation methods are done manually by radiologists and are very time-consuming, often taking several hours to complete a single patient's scan. Thus, this project proposes SmartClick, a novel interactive segmentation tool that not only significantly speeds up the currently used manual segmentation process, but also outperforms existing interactive methods in accuracy and efficiency. SmartClick employs a novel one-click paradigm where the user could segment an entire liver tumor with just one click through a recursive function. By segmenting six whole patient cases, SmartClick emerged to have a dice score percentage of 91.4% compared to the ground truth, and an average tumor segmentation time of just 8.1 seconds. This is a 5.696% increase in accuracy and 83% increase in speed compared to the previous interactive segmentation method that utilized level-set algorithms. Finally, a user-friendly and practical application designed with the QT framework was also constructed to test the efficiency of SmartClick and thus can be quickly adopted and used in a research facility.

Team - Smart Mirror @ Stonybrook COMPAS Lab

Summer between 9th and 10th Grade

CollinLi_SmartMirror_paper.pdf

A Proposal of Deep-Learning-Based Magic Mirror Modules to identify Specific Health Aspects.

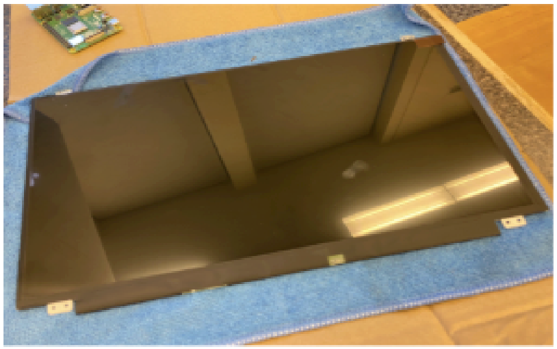
Working with two students one year older than me and under the mentorship of Professor Ferdman at Stonybrook, we brainstormed and wrote manuscripts on our proposed device. I programmed the majority of the software. The clips shown in the video were all recorded by me demonstrating my progress.

Description:

One of the most prevalent healthcare problems in the world today is that people choose not to visit the doctor's office regularly. Even when people show symptoms of sickness, they decide not to go for unsubstantial reasons, including not having enough time. Our proposed solution is to use the technique of deep learning and the advancement in IoT infrastructure to help make healthcare more accessible to the general public. I used an open-source API called the Magic Mirror, which has software capable of displaying a plethora of data. This data includes weather, calendar, news, etc. Another fundamental aspect of the Magic Mirror is that it is open-source, allowing a user to customize and display whatever they want on the screen. I used this capability to create two widgets capable of detecting eyebags and acne on a human face. Each of these modules was created using convolutional neural networks. I had three approaches to creating an eye bag module, and I separated the full face images into four sections. I found that the eye bag module performed best by cropping full faces images into just the eye region of the face. For the acne module, I also trimmed the face into four different sections, optimizing its power and identifying the area where the acne was present.

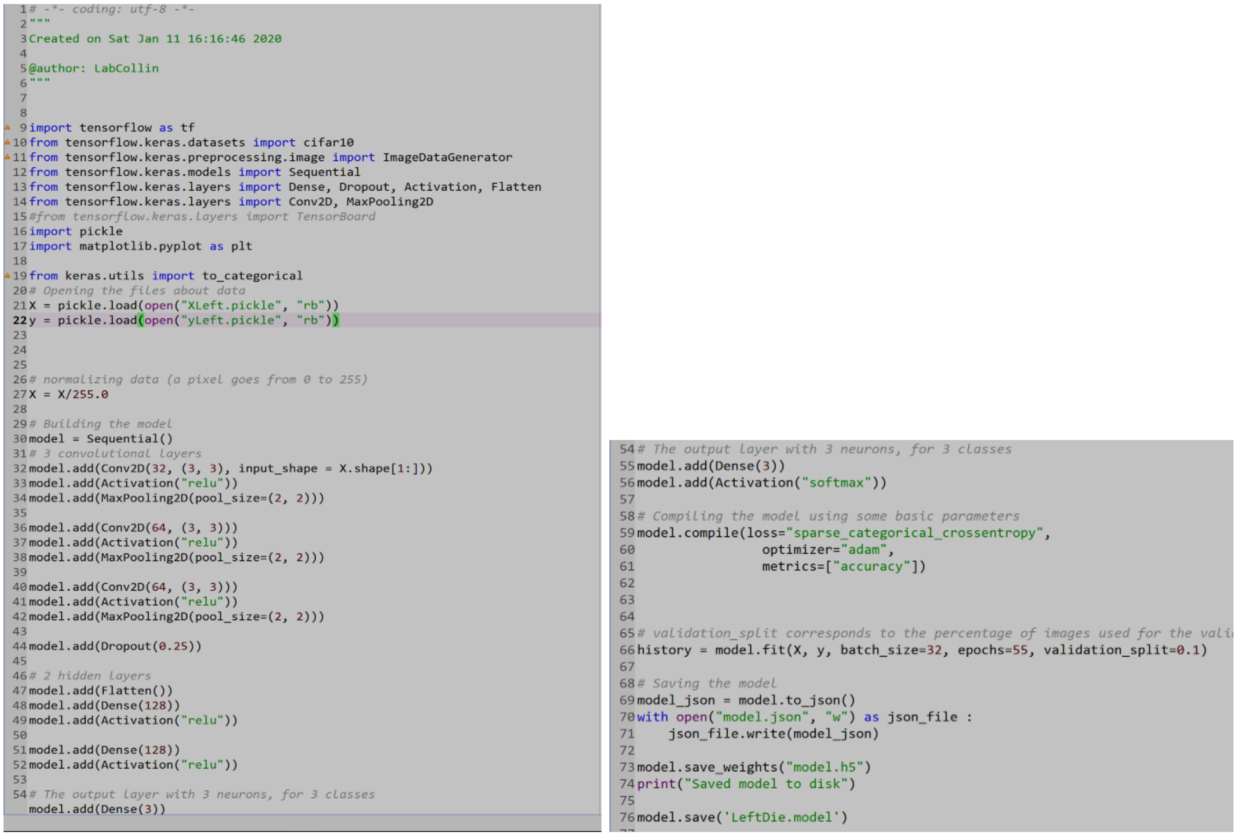
Hardware

Raspberry Pi 4, LCD Screen, Double Sided Mirror, HD USB Camera



The smart mirror was a Raspberry Pi powering a monitor behind a double-sided mirror. The Raspberry Pi would be running the open-source Magic Mirror software with additional modules that I programmed. The frontend was written in javascript and HTML and backend in python.

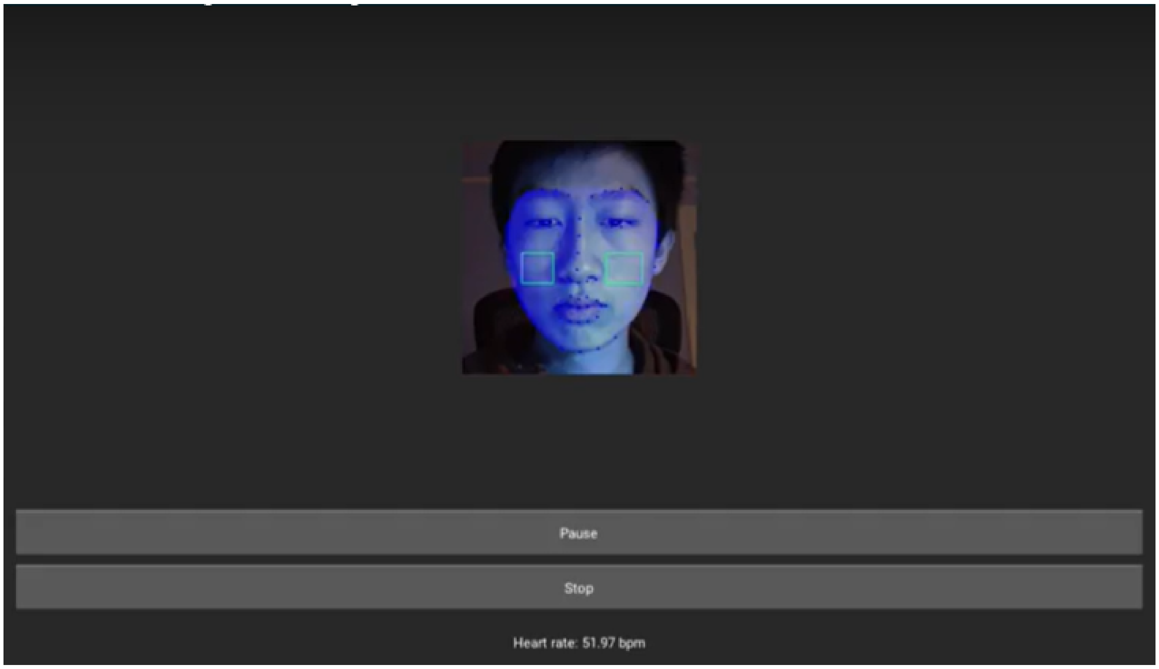
Software



```python
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 11 16:16:46 2020

@author: LabCollin
"""


import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
#from tensorflow.keras.layers import TensorBoard
import pickle
import matplotlib.pyplot as plt

from keras.utils import to_categorical
# Opening the files about data
X = pickle.load(open("XLeft.pickle", "rb"))
y = pickle.load(open("yLeft.pickle", "rb"))



# normalizing data (a pixel goes from 0 to 255)
X = X/255.0

# Building the model
model = Sequential()
# 3 convolutional layers
model.add(Conv2D(32, (3, 3), input_shape = X.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))

# 2 hidden layers
model.add(Flatten())
model.add(Dense(128))
model.add(Activation("relu"))

model.add(Dense(128))
model.add(Activation("relu"))

# The output layer with 3 neurons, for 3 classes
model.add(Dense(3))
```

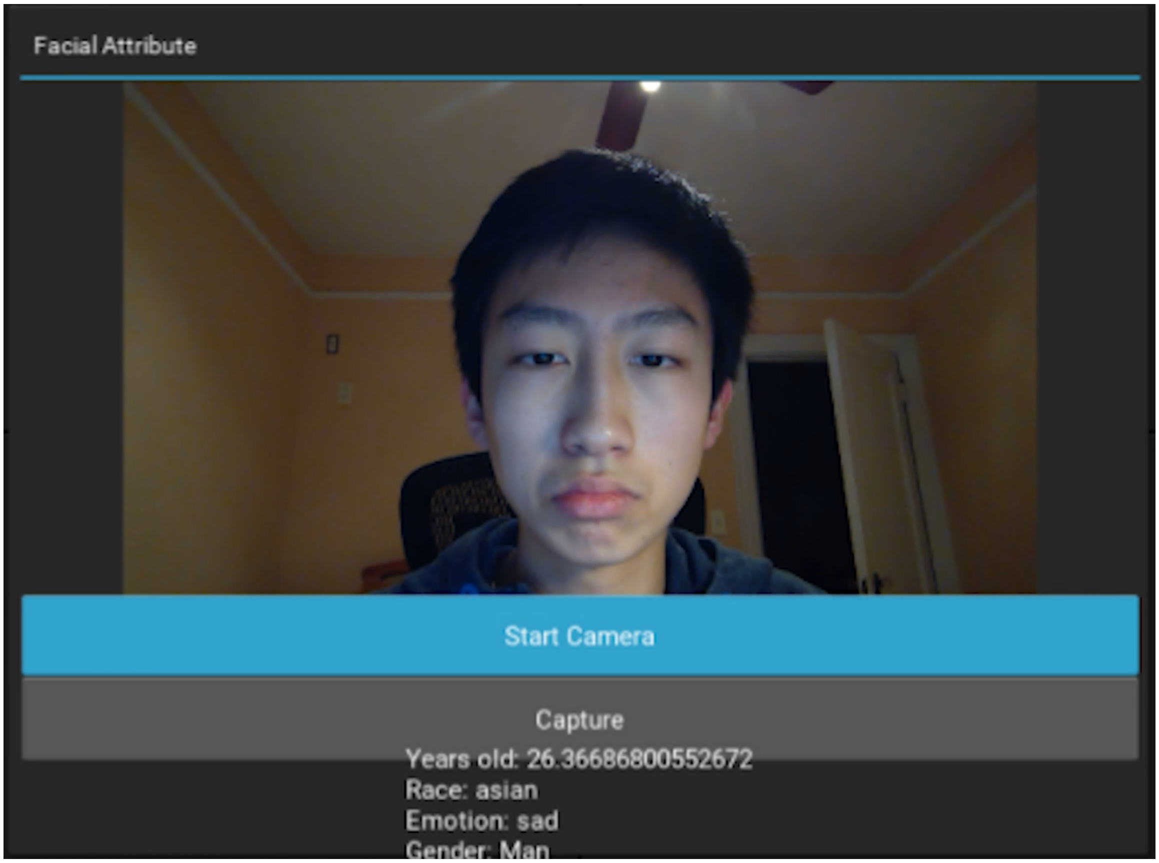
```python
# The output layer with 3 neurons, for 3 classes
model.add(Dense(3))
model.add(Activation("softmax"))

# Compiling the model using some basic parameters
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="adam",
              metrics=["accuracy"])



# validation_split corresponds to the percentage of images used for the vali
history = model.fit(X, y, batch_size=32, epochs=55, validation_split=0.1)

# Saving the model
model_json = model.to_json()
with open("model.json", "w") as json_file :
    json_file.write(model_json)

model.save_weights("model.h5")
print("Saved model to disk")

model.save('LeftDie.model')
```

Source python code for the Convolutional Neural Network to train a model that could detect peripheral puffiness/eye bags.



Heart rate: 51.97 bpm

The heart rate module was done using face Detection and region of interest tracking—first facial detection using the python library dlib and this [dat file](). The cheeks were identified, accounting for moving faces and head tilts. Finally, an Eulerian Color Magnification was performed that amplified the Green channel variations in each pixel in the ROI. Finally, a signal processing technique was performed to estimate the frequencies of green channel variations using Fourier Transform. The methodology of this was taken from [here]().



```
Years old: 26.36686800552672
Race: asian
Emotion: sad
Gender: Man
```

I'm not actually sad here, it's acting! I must have actually been very happy it worked.

Team - Object Detection for FIRST Powercells @ Great Neck South High School Team 2638 - [team website]()

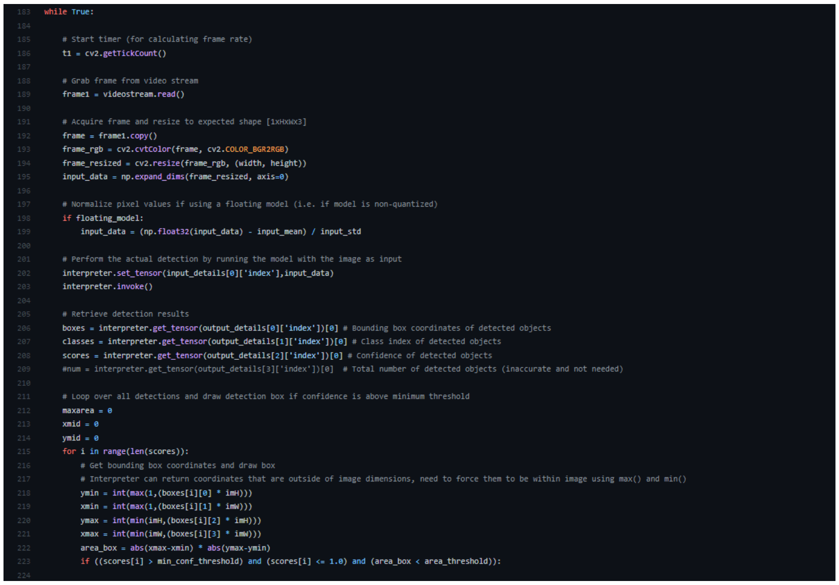
School year during 10th and 11th grade

As the programming leader of our team, we decided to ambitiously advance our robot's computer vision with deep learning approaches. We successfully trained an object detector that could locate power cells (FIRST game items) with a constant fps.

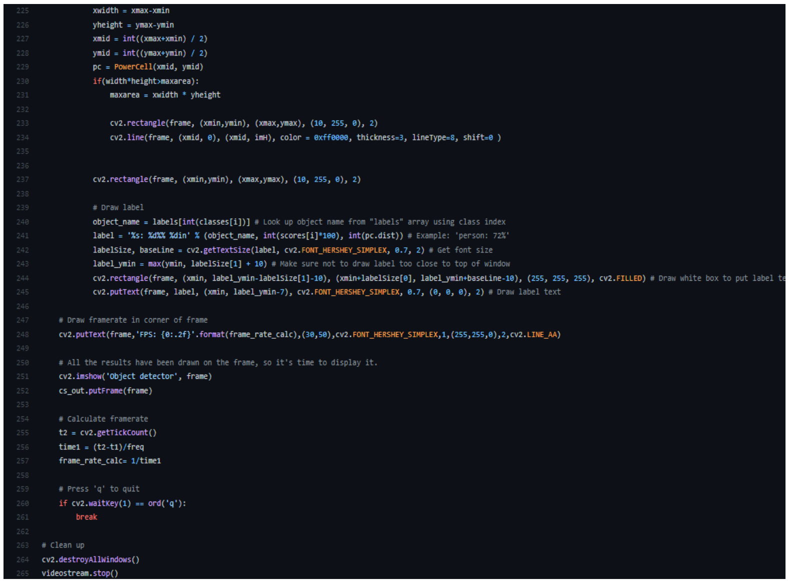
Hardware

Raspberry Pi 4, Google Coral TPU Accelerator, USB Camera

Software



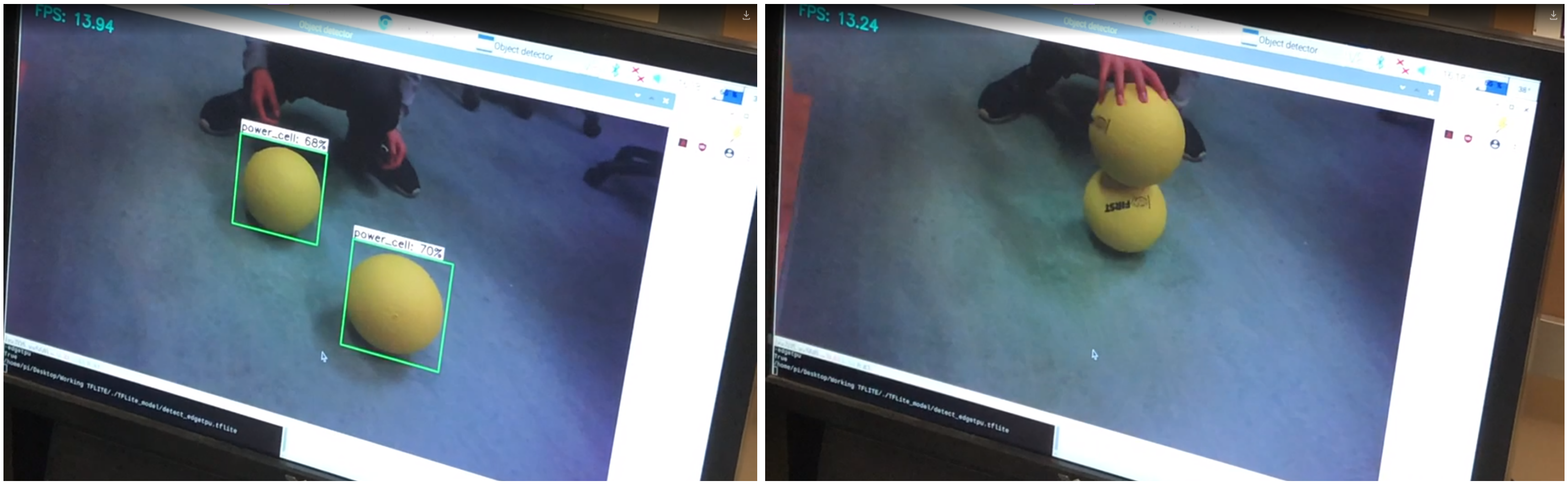
```python
while True:

    # Start timer (for calculating frame rate)
    t1 = cv2.getTickCount()

    # Grab frame from video stream
    frame1 = videostream.read()

    # Acquire frame and resize to expected shape [1xHxWx3]
    frame = frame1.copy()
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame_resized = cv2.resize(frame_rgb, (width, height))
    input_data = np.expand_dims(frame_resized, axis=0)

    # Normalize pixel values if using a floating model (i.e. if model is non-quantized)
    if floating_model:
        input_data = (np.float32(input_data) - input_mean) / input_std

    # Perform the actual detection by running the model with the image as input
    interpreter.set_tensor(input_details[0]['index'],input_data)
    interpreter.invoke()

    # Retrieve detection results
    boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box coordinates of detected objects
    classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index of detected objects
    scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence of detected objects
    #num = interpreter.get_tensor(output_details[3]['index'])[0]  # Total number of detected objects (inaccurate and not needed)

    # Loop over all detections and draw detection box if confidence is above minimum threshold
    maxarea = 0
    xmid = 0
    ymid = 0
    for i in range(len(scores)):
        # Get bounding box coordinates and draw box
        # Interpreter can return coordinates that are outside of image dimensions, need to force them to be within image using max() and min()
        ymin = int(max(1,(boxes[i][0] * imH)))
        xmin = int(max(1,(boxes[i][1] * imW)))
        ymax = int(min(imH,(boxes[i][2] * imH)))
        xmax = int(min(imW,(boxes[i][3] * imW)))
        area_box = abs(xmax-xmin) * abs(ymax-ymin)
        if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0) and (area_box < area_threshold)):

```



```python
            xwidth = xmax-xmin
            yheight = ymax-ymin
            xmid = int((xmax+xmin) / 2)
            ymid = int((ymax+ymin) / 2)
            pc = PowerCell(xmid, ymid)
            if(width*height>maxarea):
                maxarea = xwidth * yheight

                cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)
                cv2.line(frame, (xmid, 0), (xmid, imH), color = 0xff0000, thickness=3, lineType=8, shift=0 )


            cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)

            # Draw label
            object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index
            label = '%s: %d%% %din' % (object_name, int(scores[i]*100), int(pc.dist)) # Example: 'person: 72%'
            labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
            label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of window
            cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw white box to put label te
            cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text

    # Draw framerate in corner of frame
    cv2.putText(frame,'FPS: {0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),2,cv2.LINE_AA)

    # All the results have been drawn on the frame, so it's time to display it.
    cv2.imshow('Object detector', frame)
    cs_out.putFrame(frame)

    # Calculate framerate
    t2 = cv2.getTickCount()
    time1 = (t2-t1)/freq
    frame_rate_calc= 1/time1

    # Press 'q' to quit
    if cv2.waitKey(1) == ord('q'):
        break

# Clean up
cv2.destroyAllWindows()
videostream.stop()
```

Working with two other robotics programming members, I led our small group to retrain a TensorFlow Lite model to run on a Raspberry Pi 4 with Google Coral TPU accelerator. This was written in python.

Photos

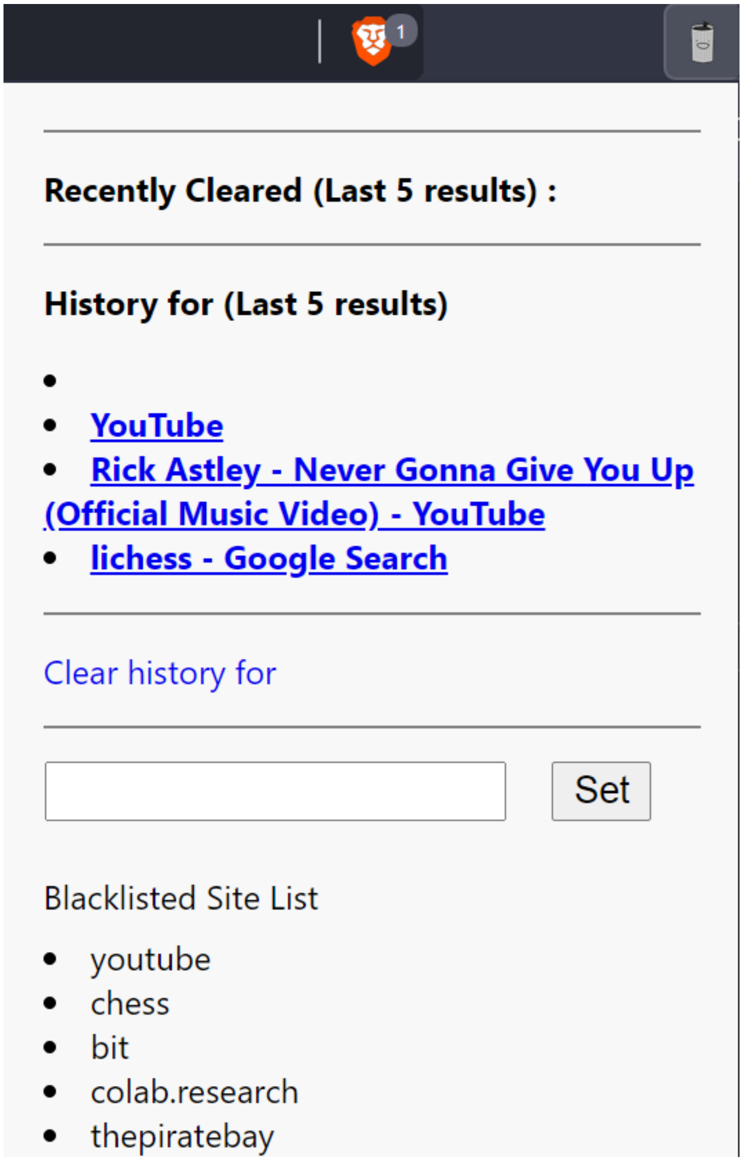


Some visual demos of the model running on a raspberry pi with google TPU coral accelerator are attached. The second photo was testing the model to its boundaries. Well, looks like we now know one limitation. (Those are my hands)

Individual - History Deleter Chrome Extension - Github link

Summer during 11th and 12th grade

Using chromium's extension API, I constructed a simple but useful extension that automatically deleted your history based on specific keywords that you could set. This was written in javascript.



Recently Cleared (Last 5 results) :

History for (Last 5 results)

-
- YouTube
- Rick Astley - Never Gonna Give You Up (Official Music Video) - YouTube
- lichess - Google Search

Clear history for

Blacklisted Site List

- youtube
- chess
- bit
- colab.research
- thepiratebay

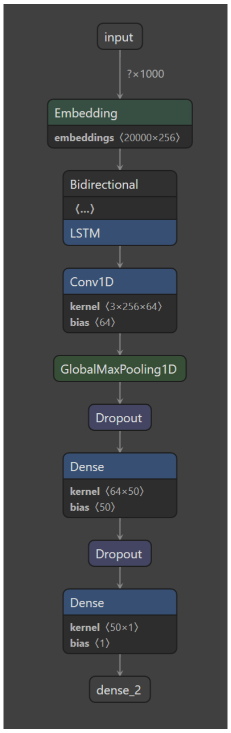
While quite a crude layout, the extension is simple and gets the job done. To add or remove keywords to blacklist sites, write your keyword into the text bar and click the button "Set". There will be no duplicate values (since it is a Set). If the keyword already exists in the set, the keyword will be removed.

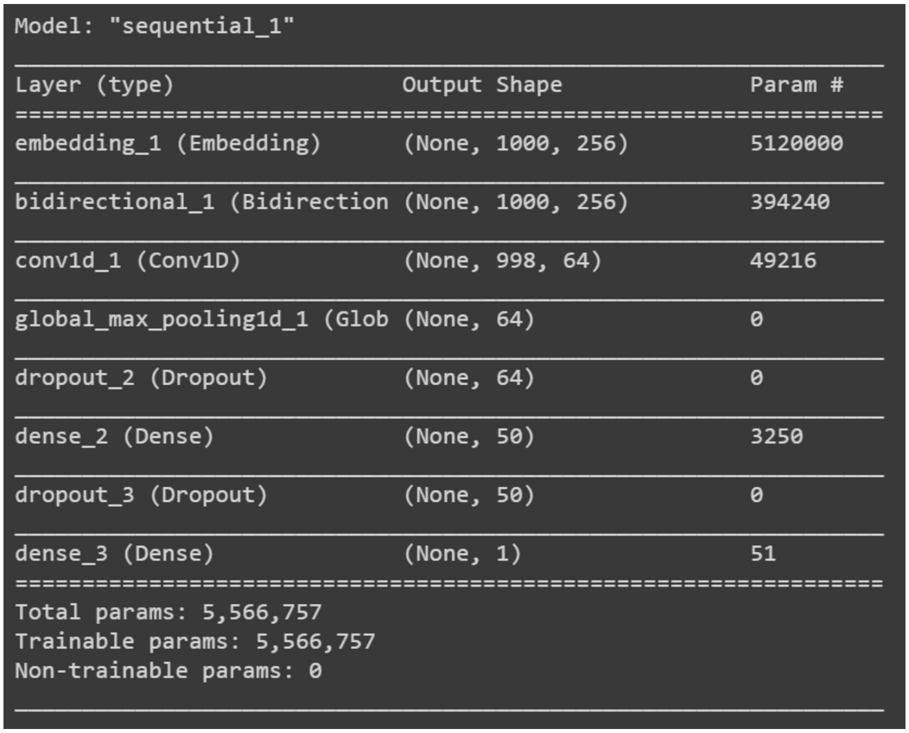
Team - Misinformation Detector Chrome Extension - GitHub link

Summer during 11th and 12th grade

CollinLi_Real-Talk-Hackaton_ppt.pdf

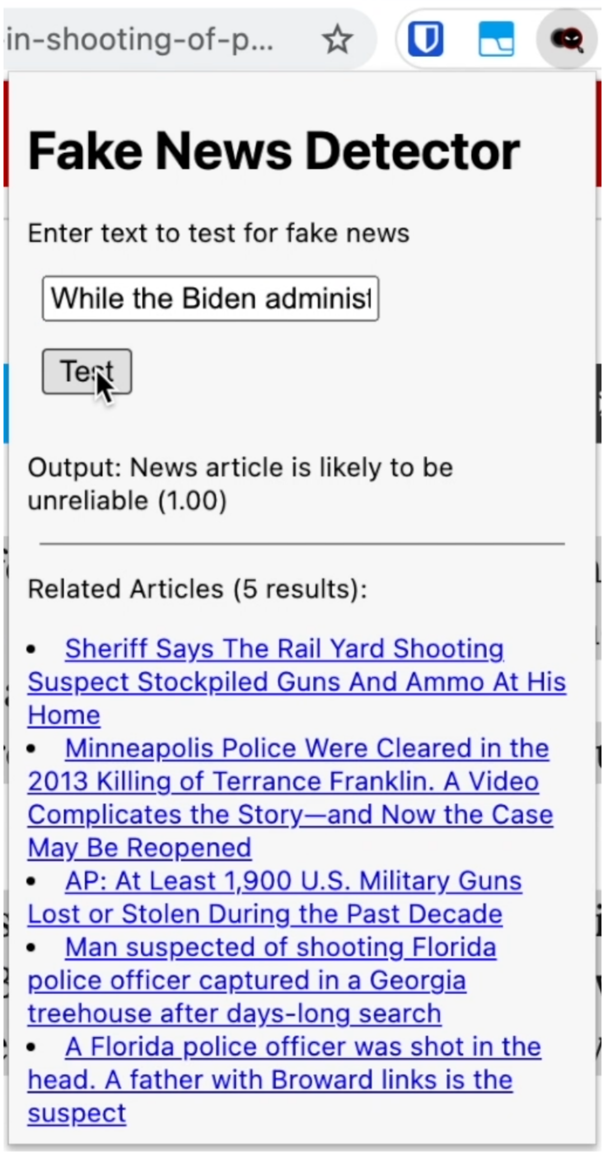
Working with one other programmer and an *aspiring* artist (my friends) who drew the extension logo and helped with the presentation, a Tensorflow for JS deep learning model was constructed to fight against misinformation. Why? Well, my grandparents and even parents are always get tricked by fake news.
This was presented at the 2021 June TeenHacks, a New York hackathon.





```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 1000, 256)         5120000
_________________________________________________________________
bidirectional_1 (Bidirection (None, 1000, 256)         394240
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 998, 64)           49216
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 64)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                3250
_________________________________________________________________
dropout_3 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 51
=================================================================
Total params: 5,566,757
Trainable params: 5,566,757
Non-trainable params: 0
_________________________________________________________________
```

The figure shows a summary of the model architecture of our convolutional neural network trained with Tensorflow and Keras. Model Architecture describes the various layers involved in the deep learning cycle and involves the significant steps being carried out in transforming raw data into training data sets capable of enabling the decision-making of a system. This was written in python.



in-shooting-of-p...

Fake News Detector

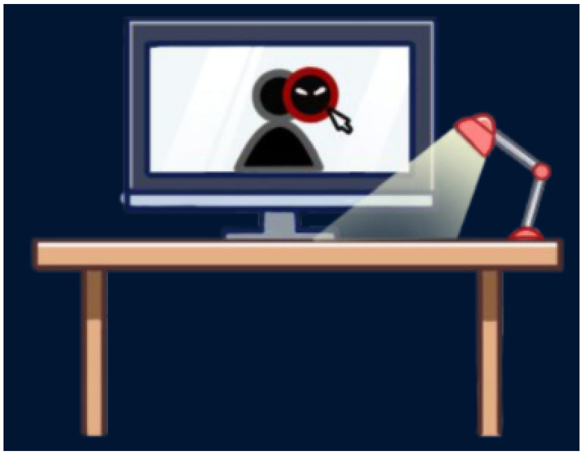
Enter text to test for fake news

While the Biden adminisi

Output: News article is likely to be unreliable (1.00)

Related Articles (5 results):

- Sheriff Says The Rail Yard Shooting Suspect Stockpiled Guns And Ammo At His Home
- Minneapolis Police Were Cleared in the 2013 Killing of Terrance Franklin. A Video Complicates the Story—and Now the Case May Be Reopened
- AP: At Least 1,900 U.S. Military Guns Lost or Stolen During the Past Decade
- Man suspected of shooting Florida police officer captured in a Georgia treehouse after days-long search
- A Florida police officer was shot in the head. A father with Broward links is the suspect

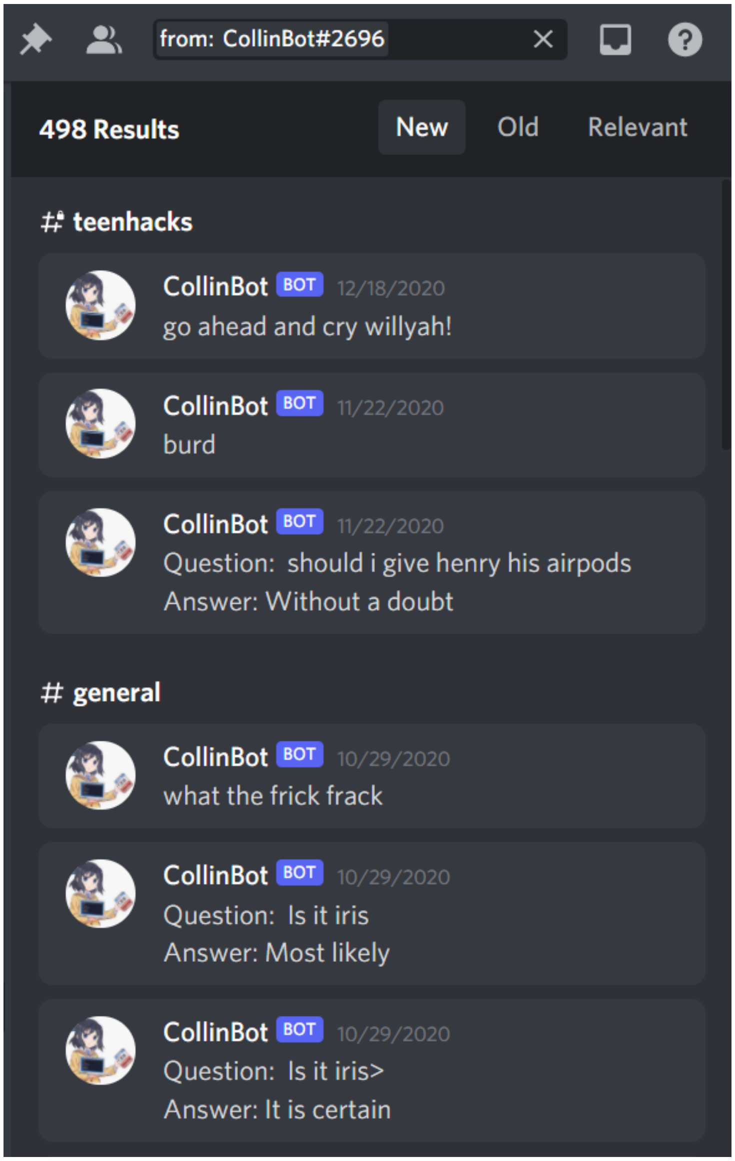


(Left) Screenshot of chrome extension working well with both deep learning detection and providing related articles (using a news API). (Right) Student-made logo for this project.

Individual - Discord Bot - School year during 11th grade

Fall of 11th grade

Using nltk sentiment analysis in Python and Discord's bot API, I constructed a program that would detect cases of "bullying" and reply with responses. Other features were included such as mentioning a catchphrase or acting as a magic 8-ball.



teenhacks

CollinBot BOT 12/18/2020
go ahead and cry willyah!

CollinBot BOT 11/22/2020
burd

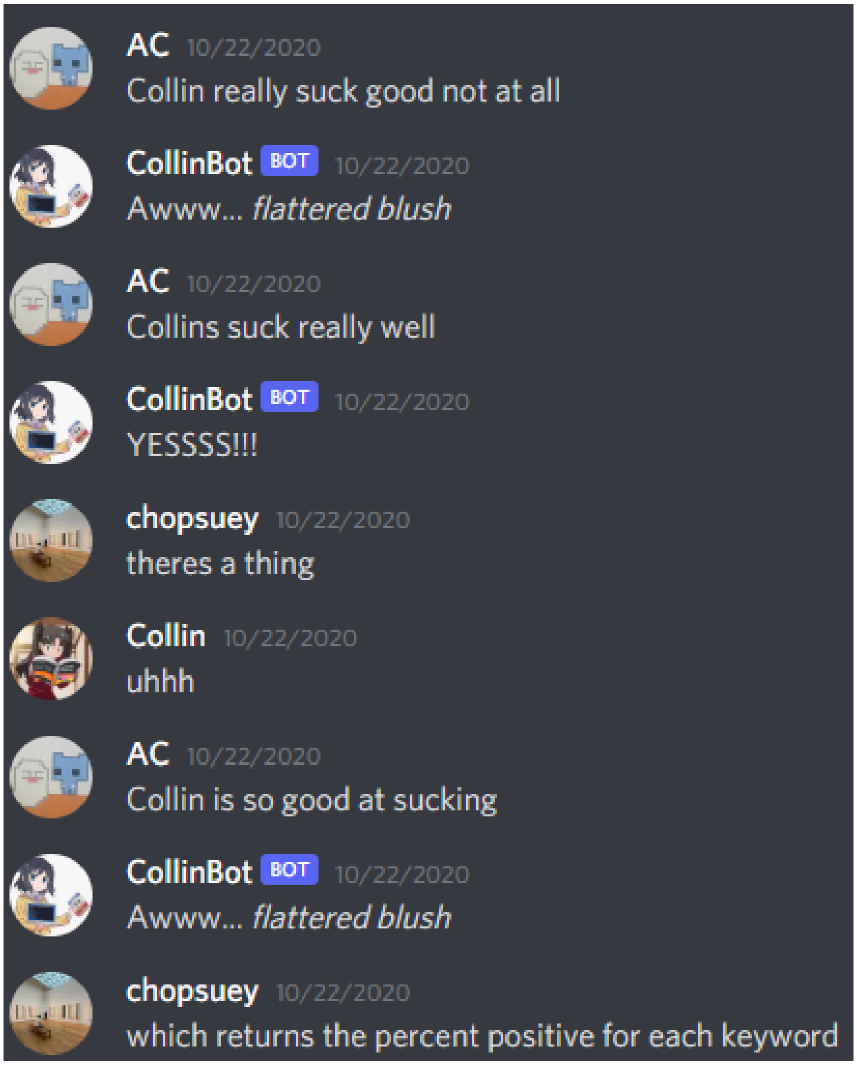
CollinBot BOT 11/22/2020
Question: should i give henry his airpods
Answer: Without a doubt

general

CollinBot BOT 10/29/2020
what the frick frack

CollinBot BOT 10/29/2020
Question: Is it iris
Answer: Most likely

CollinBot BOT 10/29/2020
Question: Is it iris>
Answer: It is certain



AC 10/22/2020
Collin really suck good not at all

CollinBot BOT 10/22/2020
Awww... *flattered blush*

AC 10/22/2020
Collins suck really well

CollinBot BOT 10/22/2020
YESSSS!!!

chopsuey 10/22/2020
theres a thing

Collin 10/22/2020
uhhh

AC 10/22/2020
Collin is so good at sucking

CollinBot BOT 10/22/2020
Awww... *flattered blush*

chopsuey 10/22/2020
which returns the percent positive for each keyword

This bot was deployed into one of my discord servers of my friends and responded almost 500 times. I also developed a feature so that if a positive thing was said, it would respond with positive phrases… However, that did not end well as my friends found loopholes. Now that I am thinking about it, was this a good idea?

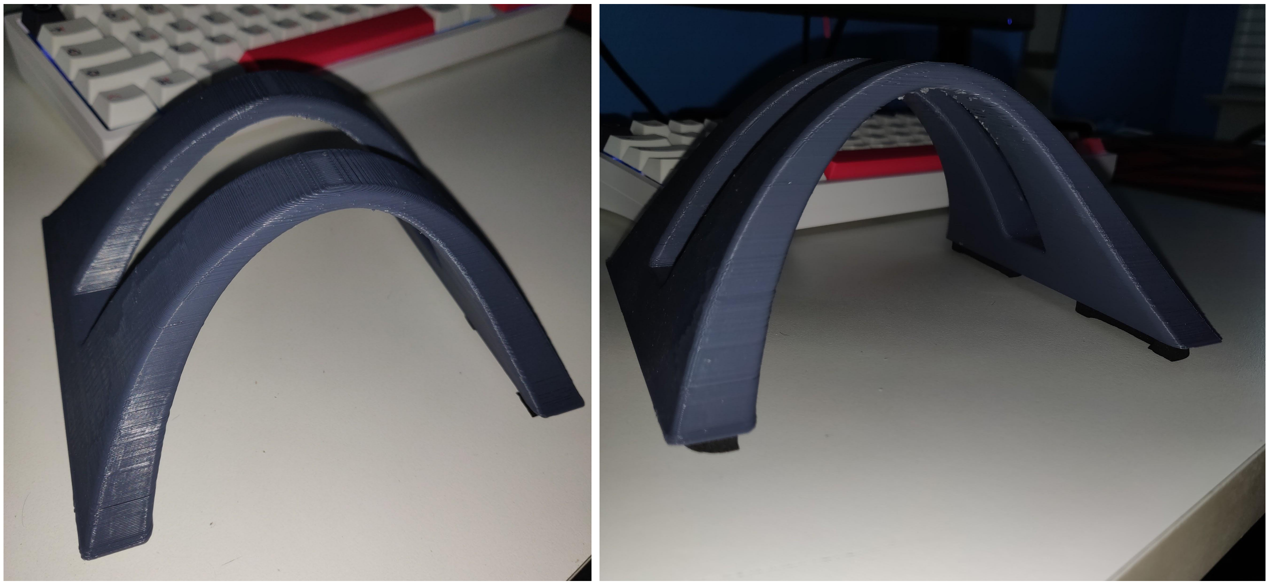
Individual - Personal Website created with NextJS (https://collinli-blog.vercel.app/)

Summer during 11th and 12th grade

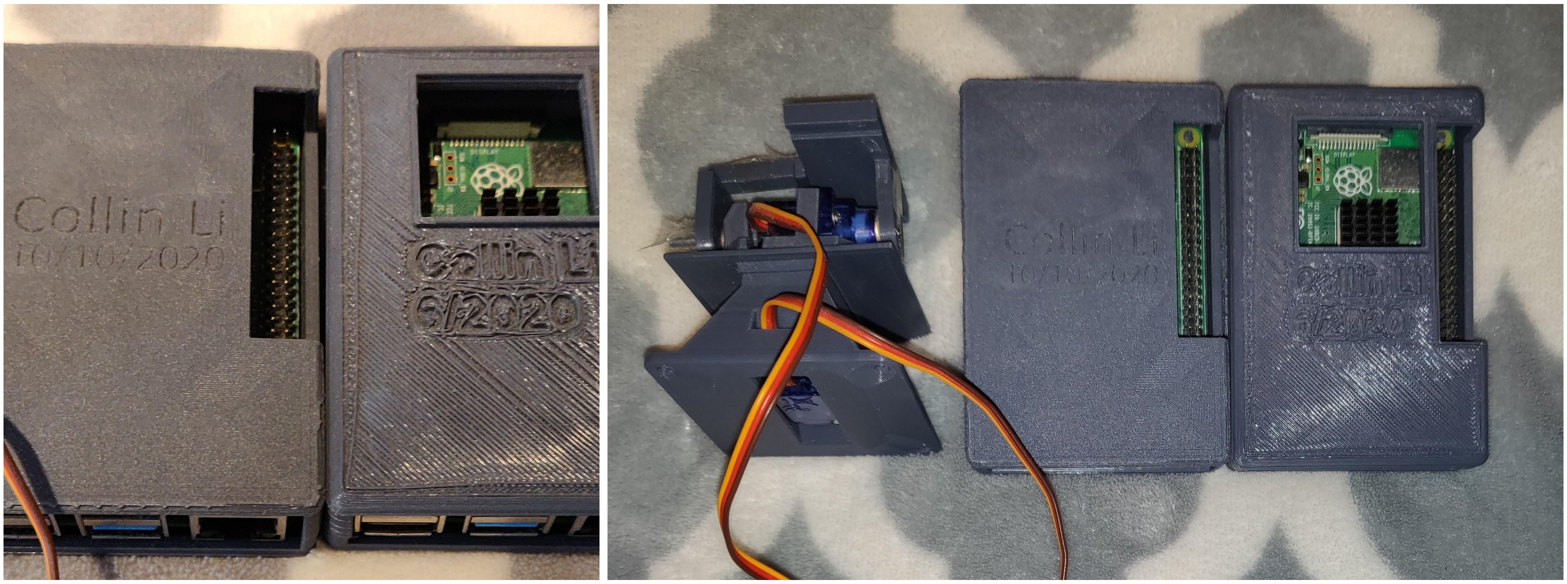
Why? Well, why not? I see this as a cool way to express myself!

Individual - 3D Printing

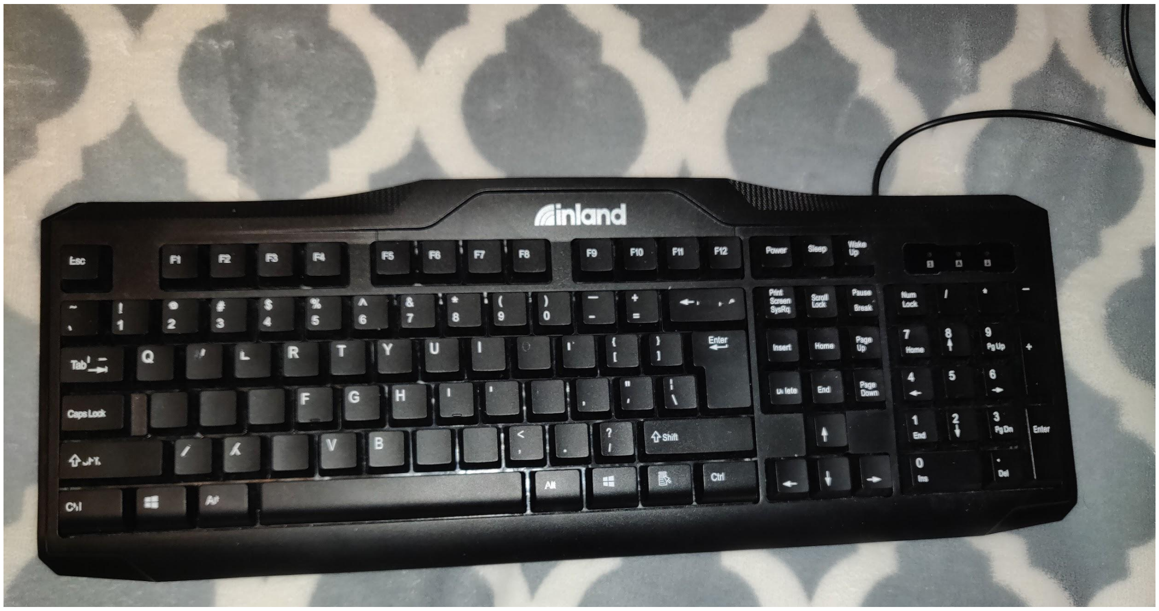
Not much here. Just some cool things I 3D modeled and printed as a hobbyist. As a Casey Neistat fan (youtube vlogger), I try to customize everything with a bit of "my touch."



A homemade 3D printed laptop stand (WITH EVEN FOAM FEET for traction) for my totally very very very powerful (2 core CPU) laptop that is passively cooled. (Oh and that's my brother's mechanical keyboard… mine is a bit more different…) ~~I even downloaded some extra ram!~~



3D printed raspberry pi cases! I have dozens of these, many of which failed horrendously due to poor printer settings. Oh and that thing on the left is a 2 axis gimbal



Oh yeah… here is my keyboard. At a whopping $3 + tax, it has served me well along all of these projects. Funny enough, some of the key marks are disappearing and in my opinion, a completely black keyboard would look pretty cool (although not so practical).