Part 2 - Technical part research

week1

- Application of machine learning based on gesture interaction in sports.

Reference - https://vimeo.com/76664145

- Gesture Recognition
- Interactive machine learning
- < Sketch Band >

Sketch Band is a machine learning music studio that was developed to allow people to draw in- struments and play them with household objects in a fun and understandable way. It's simple to use. First, individuals can set up the machine learning software and find a pen, paper, and something to play their instrument with. Second, they can draw an instrument (the machine learn- ing software was trained to know a few – drums, guitar, symbol). Lastly, you can play the instrument by placing the drawing in front of the camera and hearing the sound it produces.

Reference - https://vimeo.com/430569989

week2

- Tools: Wekinator, MIMIC, GestureFollower, Teachable Machine， ml4a， ...
- Wekinator:

- Creation of gesturally-controlled animations and games

Control interactive visual environments created in Processing, OpenFrameworks, or Quartz Composer, or

game engines like Unity, using gestures sensed from webcam, Kinect, Arduino, etc.

- Creation of systems for gesture analysis and feedback

Build classifiers to detect which gesture a user is performing. Use the identified gesture to control the

computer or to inform the user how he's doing.

http://www.wekinator.org/

- Dynamic time warping in Wekinator with mouse:

https://www.youtube.com/watch?v=J4viXTThDTE

week3

- Hand pose information from a webcam in realtime

Training a Neural Network to Detect Gestures with OpenCV in Python

https://towardsdatascience.com/training-a-neural-network-to-detect-gestures-with-opencv-

in-python-e09b0a12bdf1

- Handtrack.js: Hand Tracking Interactions in the Browser using Tensorflow.js and 3 lines of code.

https://towardsdatascience.com/handtrackjs-677c29c1d585

```
<script src="https://cdn.jsdelivr.net/npm/handtrackjs/dist/handtrack.min.js"></script>
const img = document.getElementById('img');
handTrack.load().then(model => {
  model.detect(img).then(predictions => {
    console.log('Predictions: ', predictions) // bbox 预测
  });
});
```

week4

- Handtrack.js Official website

https://victordibia.com/handtrack.js/#/

- Handtrack.js Prototype site

https://codepen.io/victordibia/pen/RdWbEY

- left and right hand

https://codepen.io/md-azeem/pen/xxEZbrz

- When should use handtrack.js

  Some (not all) relevant scenarios are listed below:

- When mouse motion can be mapped to hand motion for control purposes.
- When an overlap of hand and other objects can represent meaningful interaction signals (e.g a touch
  or selection event for an object).
- Scenarios where the human hand motion can be a proxy for activity recognition (e.g. automatically
  tracking movement activity from a video or images of individuals playing chess, or tracking a persons
  golf swing). Or simply counting how many humans are present in an image or video frame.
- Interactive art installations. Could be a fun set of controls for interactive art installations.
- Teaching others about ML/AI. The handtrack.js libary provides a valuable interface to demonstrate
  how changes in the model parameters (confidence threshold, IoU threshold, image size etc) can
  affect detection results.
- You want an accessible demonstration that anyone can easily run or tryout with minimal setup.

https://github.com/sannalin/API-Lab-group-3/tree/main/Carlo%20Sicanica/Example%201

week5

- fingerpose: Finger pose classifier for hand landmarks detected by TensorFlow.js' handpose model.

  https://github.com/andypotato/fingerpose

Gesture detection works in three steps:

1. Detect the hand landmarks inside the video picture
2. Estimating the direction and curl of each individual finger
3. Comparing the result to a set of gesture descriptions

Advantages:

1. A good way to quickly integrate hand detection
2. You can use a webcam to track your hands and their movements
3. It can also be used to track hands in videos or even pictures

Disadvantages:

Can't do too much behind the scenes to change the forecasting model

Simple hand detection application:

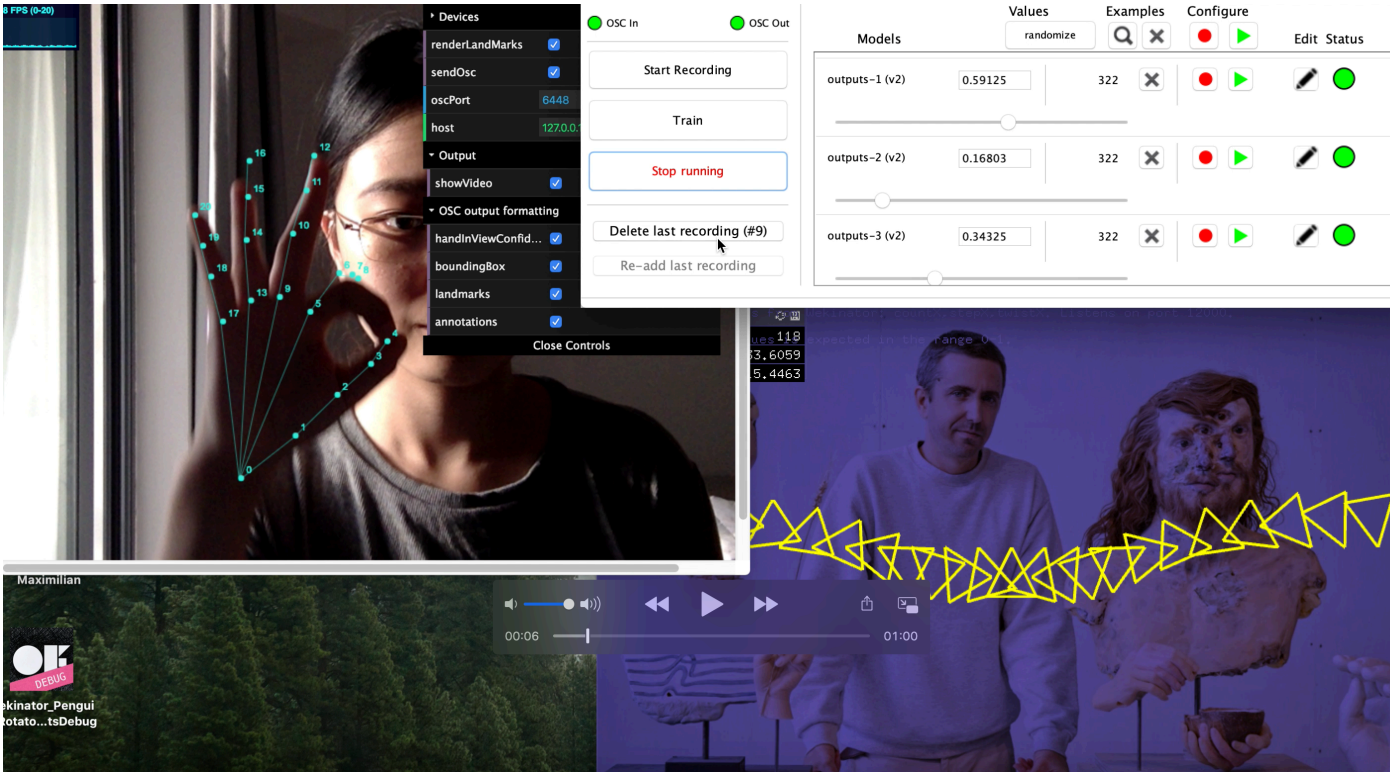
https://heartbeat.fritz.ai/introduction-to-hand-detection-in-the-browser-with-handtrack-js-and-tensorflow-e4256fa8184b

week6

- Handpose OSC

This program takes the webcam as input and passes the image stream through a neural network, which estimates the location and landmarks of a single hand and outputs the results as OSC, which is a network protocol used for connecting different programs.
The default output of HandPose OSC is 8008.

Hand gesture recognition using HandPose-OSC and Wekinator
https://fablab.ruc.dk/hand-gesture-recognition-using-handpose-osc-and-wekinator/



week7

- Wekinator

Wekinator is a piece of software which has OSC as input and output. The user can demonstrate inputs and their desired outputs and then train a machine learning algorithm to a predict the desired output given an input.

- Using Wekinator to explore supervised learning classifiers：

https://www.youtube.com/watch?v=msHPAcpAjCc

- ml4a @ itp nyu :: 02 applications, wekinator：

https://www.youtube.com/watch?v=lKmkt5LEDS8

- Wekinator will take landmarks from the different hand gestures and output the predicted gesture.

openFrameworks (C++, animation)

- VERY simple colour change (just adjusts hue)
  - Source code
- Simple 3D penguin rotation (based on ofxAssimpModelLoader example)
  - Source code
  - Mac executable **Note: If you don't see the penguin when you run it, drag the app to a new location (e.g., your desktop) then back. Make sure the app and the data/ directory share the same parent directory. Run it again.** (This is due to a peculiarity in security settings on OS X 10.12 and higher.)

I download very simple color change openframework file as output. I need to figure out how wekinator and openframework are connected together.

week8

- Openframework：Brushes from Basic Shapes

https://openframeworks.cc/ofBook/chapters/intro_to_graphics.html

Single Rectangle Brush: Using the Mouse

```
if (ofGetMousePressed(OF_MOUSE_BUTTON_LEFT)) {  // If the left mouse
  ofSetColor(255);
  ofSetRectMode(OF_RECTMODE_CENTER);
  ofDrawRectangle(ofGetMouseX(), ofGetMouseY(), 50, 50);
 // Draw a 50 x 50 rect centered over the mouse
}
```