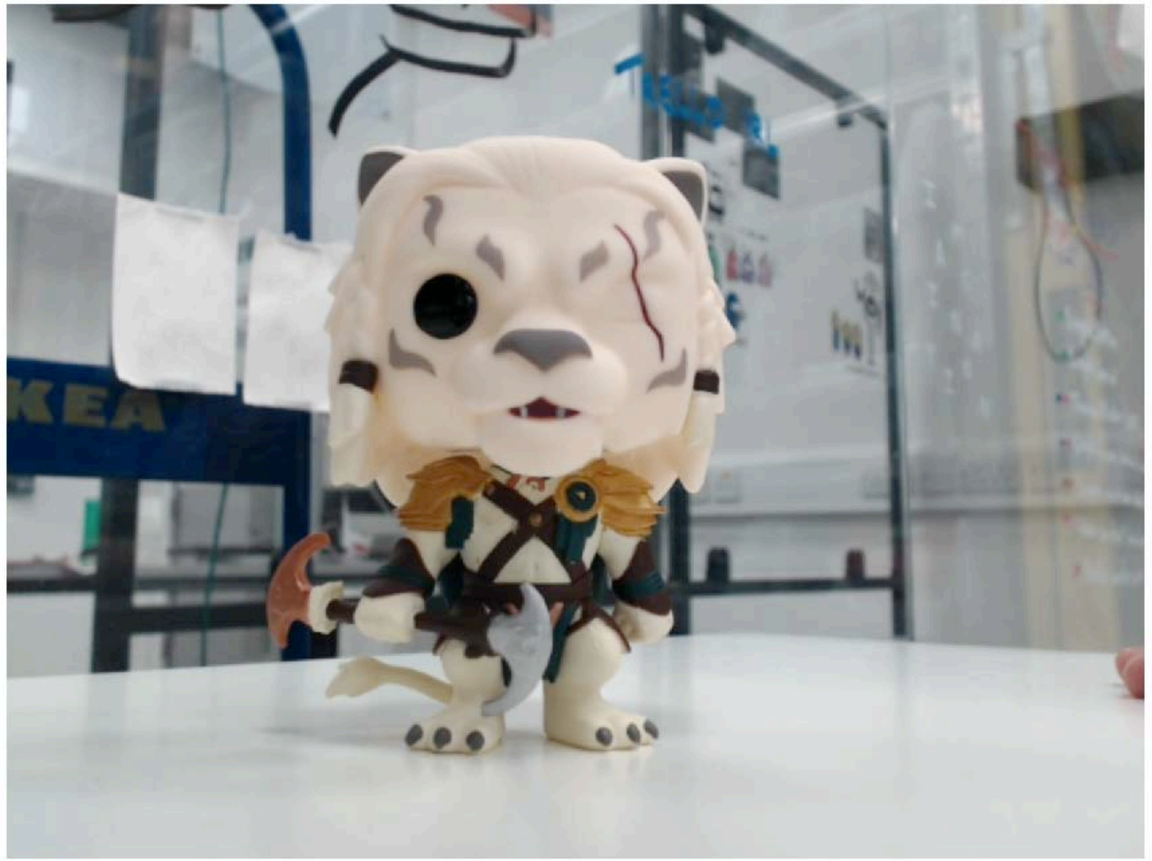
Introduction to Machine Learning

in ML5.js

Veera Jussila / CTL / 18 Jan 2022

Agenda

- What is machine learning (ML)?
- Creative uses of ML
- What is ML5?
- Coding together!
    - Image classifier
    - Skeleton tracking



toyshop 0.40

Machine Learning = Predicting

- We train a bunch of algorithms = model to make **predictions** based on data
- Example 1 : After seeing 1000 cats, model is 99% sure when it sees a cat again (prediction w/ 99% confidence)
- Example 2 : After learning the whole “Emma”, model predicts and prints a short text that Jane Austen would write

ML requires a dataset

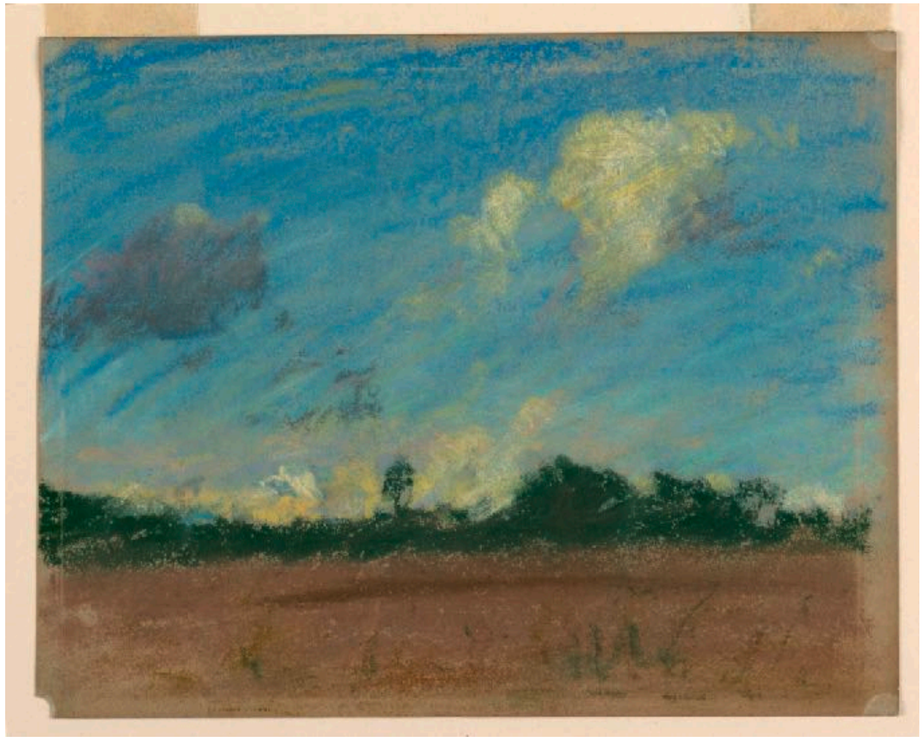
- For the model, **dataset IS the world** that it imitates
- *Classification*: mapping inputs to outputs with class labels
    - Fox is a cat with 70% confidence, a dog with 30% confidence (if no class for foxes)
- *Generative models*: repeating patterns
    - ”Learning the features of data and simplifying data representations for the purpose of finding patterns” (Tiu 2020)
    - Writing “almost like Austen”

ML can be practical…

- Trigger a sound when a human face is detected
- Open a cat flap when a cat approaches
- Play a song when somebody says the name of the song

…Or more experimental

- Make a photo look like the painting on the right ->
- Collect 100 dreams from people, print out an “average” dream
- Create imaginary dog breeds
- ?

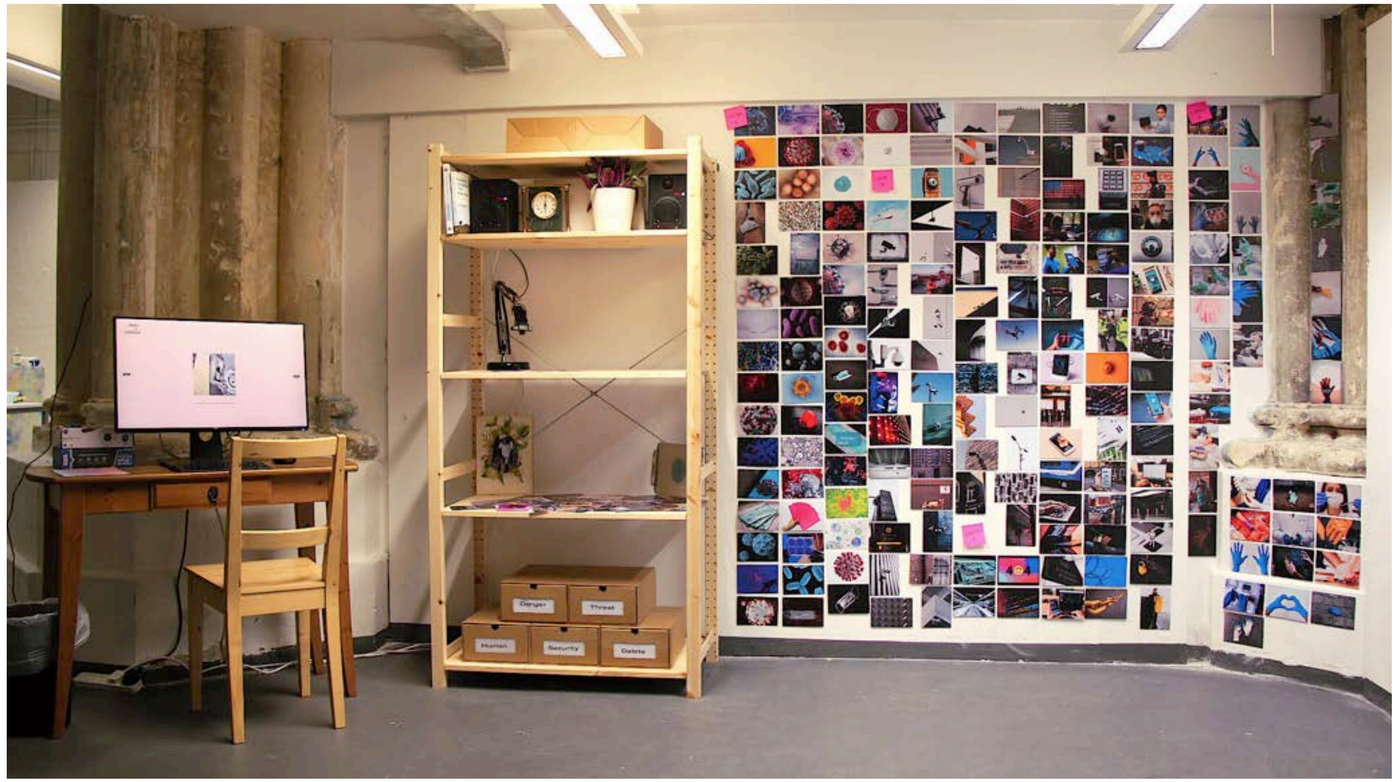


Example 1: Browser

- Hand recognition to control visuals
- Pinch to Zoom by Tom-Lucas Säger (2021)

Example 2: Installation

- Museum of Borderlands (2020)
- I used image classification to trigger sounds
- ML happens in web app, sends messages to MaxMSP



ML5.js

- Built on top of Tensorflow.js
- Both are Javascript libraries for ML projects
- ML used to be a Python thing
- Javascript becoming popular as well: ML running on browser

Popular ML5.js models

- ImageClassifier
- PoseNet
- Facemesh / FaceAPI

Custom models

- Require **custom datasets**
- Why to make one?
  - **More control** in creative projects
  - **More insight** into ML: it’s not magic!
  - Also: old, standard models can be problematic - be careful!
    - ML5 recently disabled word2vec - see the Twitter thread

Today we work with ready models tho

- Quick start
- Sometimes you just want to recognise a dog!
- 1. **ImageClassifier** and
  2. **PoseNet**

Step 1: Include P5.js and ML5.js in index.html

```
<head>
    <meta charset="utf-8" />
    <title>My test project</title>
    <script src="https://cdn.jsdelivr.net/npm/p5@1.4.0/lib/p5.js"></script>
    <script src="https://unpkg.com/ml5@latest/dist/ml5.min.js"></script>
</head>
```

Step 2: Change <h1> to something fitting

```
    <body>
        <h1>My ML tests!</h1>
        <script src="sketch.js"></script>
    </body>
</html>
```

Step 3: Adding variables

```
let classifier;
let video;
let resultsP;
```

- For the **ML model**, **video** and **results**
- Original code: ML5.js Github

Step 4: Creating setup

```
function setup() {
  noCanvas();
  video = createCapture(VIDEO);
  classifier = ml5.imageClassifier('MobileNet', video, modelReady);
  resultsP = createP('Loading model and video...');
}
```

- No need for canvas in this project
- Creating webcam input
- Setting “classifier” to be ML5 imageClassifier with Mobilenet, video as second argument, finally call a function called modelReady
- Variable resultsP creates a text snippet

Step 5: modelReady helper function

```
function modelReady() {
  console.log('Model Ready');
  classifyVideo();
}
```

- Print to console when this function runs - good for debugging
- Next, trigger function called classifyVideo (we'll create it next!)

Step 6: Actual classification

```
function classifyVideo() {
  classifier.classify();
}
```

- Get a prediction = ML classification for the current video frame
- Remember that in setup we have set the model to use video as its input. No need to write anything about video here

Step 7: Refresh the page

- Why don’t we see any results?

Step 8: Adding a gotResult function to show results

```
function classifyVideo() {
  classifier.classify(gotResult);
}
```

Writing the gotResult function

```
function gotResult(err, results) {
  resultsP.html(`${results[0].label  } ${nf(results[0].confidence, 0, 2)}`);
  classifyVideo();
}
```

- Trigger the function with two arguments: either we get error or results
- Our old resultsP variable gets filled with new text
- ` is for outputting a string, ${ extracts data from our results array (see next slide)
- The results array is ordered by confidence, starting from index 0
- **[0] is thus our top result** and we print its label + confidence
- Finally, call classifyVideo again

Raw data looks like this

```
▼ (3) [{…}, {…}, {…}] i
  ▶ 0: {label: 'vending machine', confidence: 0.192215248942375…
  ▶ 1: {label: 'refrigerator, icebox', confidence: 0.0530274212…
  ▶ 2: {label: 'monitor', confidence: 0.040555596351623535}
    length: 3
```

You should have a video with live classifications now!

mask 0.20

Questions? And…Break!

Part 2: Working with PoseNet

Step 1: Save current progress to a separate file

- **Rename** sketch.js to something else, like sketch2.js or sketchImageClassifier.js
- Make a brand new file **sketch.js** - now our html refers to this new file!
- We can now freely edit sketch.js to **include PoseNet** and not have imageClassifier there confusing us

Step 2: Add text paragraph to html file

```
<body>
    <h1>My ML tests!</h1>
    <p id='status'>Loading model...</p>
    <script src="sketch.js"></script>
</body>
```

Step 3: New variables to sketch.js

```
let video;
let poseNet;
let poses = [];
```

Step 4: Setup

```
function setup() {
  createCanvas(640, 480);
  video = createCapture(VIDEO);
  video.size(width, height);
  poseNet = ml5.poseNet(video, modelReady);
  poseNet.on("pose", function(results) {
    poses = results;
  });
  video.hide();
}
```

- Variable poseNet is filled with ML5 posenet model. Once done, trigger modelReady function
- Triggers an event that fills variable “poses” with a stream of number results coming from our model
- Hide the actual video input

Step 5: adding modelReady function

```
function modelReady() {
    select("#status").html("Model Loaded");
}
```

- We did this earlier as well!
- When model is ready, text in the html text paragraph will change to: Model loaded

Try to run it!

- Model is loaded, webcam is on…
- But we are **not using ML results** or drawing anything to canvas!

Step 6: Draw function

```
function draw() {
  image(video, 0, 0, width, height);
  drawSkeleton();
}
```

- Running constantly - up to 60 times per second
- Draw video frames
- Trigger drawSkeleton function - let’s write it next

Step 7: Tracking skeletons

- PoseNet can detect a person or multiple people
- **Loads of possibilities:**
  - **Change visuals when two people are in the room**
  - Trigger a sound when somebody raises their hand
  - Make an animation that follows a dancer

drawSkeleton function

```
function drawSkeleton() {
  for (let i = 0; i < poses.length; i += 1) {
    const skeleton = poses[i].skeleton;
    for (let j = 0; j < skeleton.length; j += 1) {
      const partA = skeleton[j][0];
      const partB = skeleton[j][1];
      stroke(255, 0, 0);
      line(partA.position.x, partA.position.y, partB.position.x, partB.position.y);
    }
  }
}
```

- Remember, poses is a long queue of number data coming from our ML model
- poses.length is the number of poses = people detected
- Each pose includes a skeleton, nose etc. Loop through all of them
- Each skeleton holds number coordinates for different body parts that are connected (knee -ankle etc). Loop through all of them
- Draw lines between x&y coordinates of body parts

Let’s wait until everybody can see a skeleton!

(Final) Step 8: Interaction

```
let video;
let poseNet;
let poses = [];
let c;
```

- Let’s add something that can change colour
- First, add a new variable called c. This is for the changing colour

Adding a rectangle with changing colour

We add these lines to the drawSkeleton() function

```
const skeleton = poses[i].skeleton;

if (skeleton.length > 1){
  c = color(255, 0, 0);
}
else {
 c = color(55, 204, 0);
}

noStroke();
fill (c);
rect(0, 0, 100, 100);
```

- If more than one skeleton, make c red
- In other cases, make c green

Want to get more specific?

- Check ML5 PoseNet example and this Shiffman tutorial for how to get information about body parts
- leftArm, rightLeg…
- They are called **keyPoints**
- **Loads of possibilities!**
  - Change sound when somebody turns their head (follow their nose, for example)

Many tools for ML!

- **RunwayML**: Software with **no coding**
- **ML5.js**: Friendly, optimised for **web projects**
- **Python**: Offline, installations, **deep learning**
- Come and ask me if in doubt :)