

**Massachusetts Institute of Technology**  
**Dept. of Electrical Engineering and Computer Science**  
**Fall Semester, 2018**  
[MIT 6.S198: Deep Learning Practicum](#)

## Assignment 1: Teachable machine

**Task 1:** Run the demo again, this time using the Chrome developer tools to watch the console window. Each time an image is classified, the console prints the number of the top 20 closest images contained in each class, and also the resulting confidence for each class. Open the file WebcamClassifier.js and find the function onSpecialButtonClick() near the top of the file. Edit the function so that it uses the Javascript window.alert command to show the same information that is printed to the console on each cycle. Now when you press the special button the program will pause and you can examine the information in the alert. Processing will resume when you press OK.

**Action 1:** The task was to return the same results of console.log to the special button. The first step in this task was to find out what was printed by console.log(). Many things are printed but the counts and confidence intervals were set near line 385, inside animate(). I took the input to console.log and copied them to global variables.

```
// Change the two NULLs below to get the global values that can be used in onSpecialButtonClick
globConf = 'Confidence for which the image matches each class: ' + confidences;
globNCounts = 'Number of the top ' + TOPK + ' closest matches in each class: ' + nCounts;

console.log('Number of the top ' + TOPK + ' closest matches in each class: ' + nCounts);
console.log('Confidence for which the image matches each class: ' + confidences);
```

From there I needed to make the alerts popup so I used window.alert to print both lines, separated a line break.

```
function onSpecialButtonClick() {
  //FILL IN YOUR SPECIAL FUNCTION
  window.alert(globNCounts + '\n' + globConf);
}
```

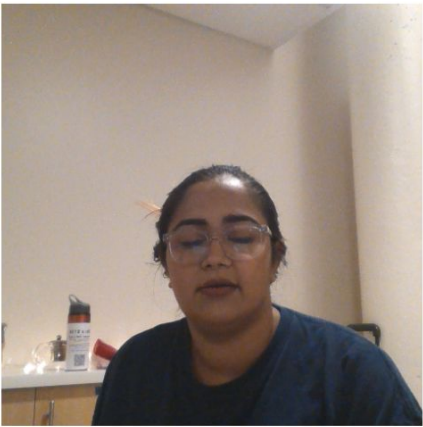
---

**Task 2:** Redo some experiments like the ones you did in section 2. Briefly write up your observations, using the confidences and top K values to support what you observe.

**Task 2-1:** Train classes with images of your face and your partner's face where you show different expressions. For example, the first class would be partner A's face making all sorts of expressions and the second class would be partner B's face making all sorts of expressions. Can the network tell the difference between the two faces? If so, how confident is it? Note that the program will start training for a class with images from the camera when you press the corresponding button. It will keep training as long as the button is depressed.

**Action 2-1:** This was a easy one. Shohini and I trained the program on our faces with each making make different faces and it was quite accurate in identifying who we were.


INPUT



Number of the top 20 closest matches in each class: 0,20,0  
Confidence for which the image matches each class: 0,1,0

OK

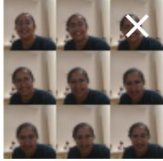
75 EXAMPLES



CONFIDENCE

TRAIN GREEN

53 EXAMPLES

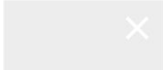


CONFIDENCE

100%

TRAIN PURPLE

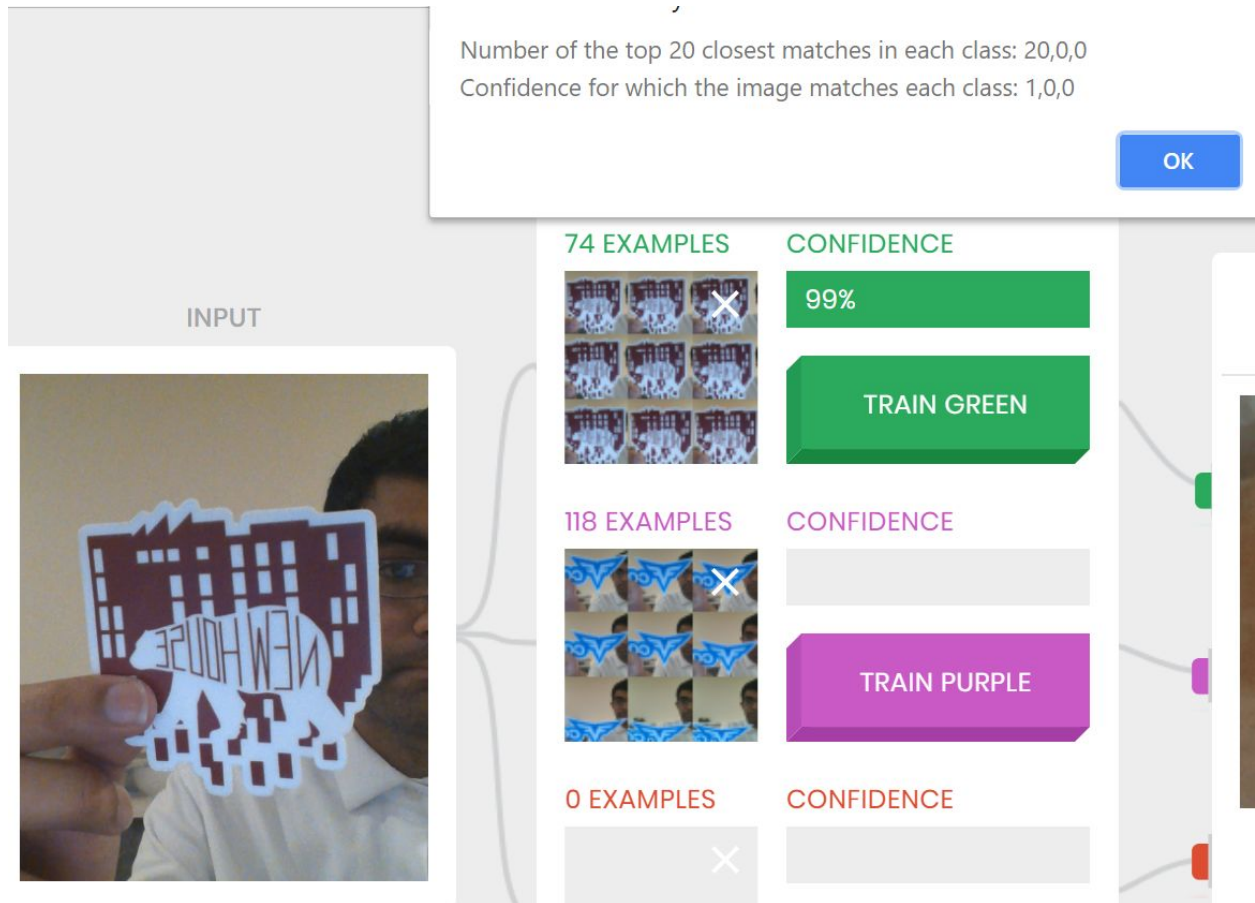
0 EXAMPLES



CONFIDENCE

**Task 2-2:** Try training on various inanimate objects or even photos on your phone that you hold up in front of the computer camera.

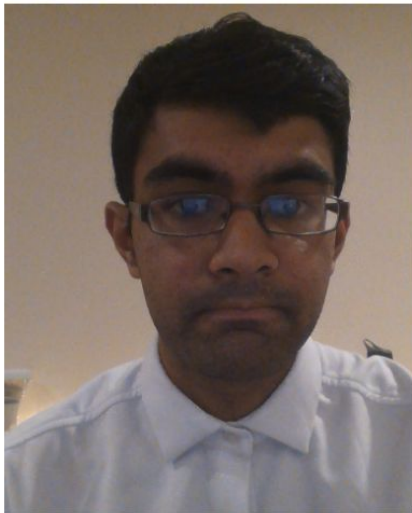
**Action 2-2:** This was also a simple one. I trained the program on two different stickers and it was very accurate.



**Task 2-3:** Train two different classes with the same image set (for example, your face in a fixed expression).

**Action 2-3:** Here the program was just very confused and it would swing wildly between 70% and 30% confident for both green and purple.


INPUT



Number of the top 20 closest matches in each class: 12,8,0  
Confidence for which the image matches each class: 0.6,0.4,0

OK

36 EXAMPLES




CONFIDENCE

46%

TRAIN GREEN

39 EXAMPLES

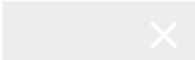


CONFIDENCE

53%

TRAIN PURPLE

0 EXAMPLES

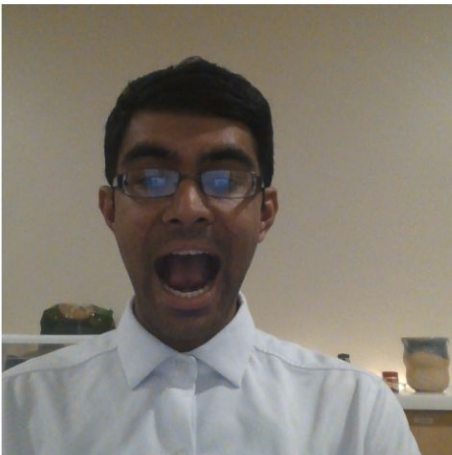


CONFIDENCE

**Task 2-4:** Train one class on two people with expression A and another class on two people with expression B. Does the network end up learning the difference between the two faces, or the difference between the two expressions? Or is it just confused (i.e., wrong, or not very confident)?

**Action 2-4:** This was very difficult for the program. It could do a better than pure chance, but overall it really struggled to correctly identify our expressions. Mostly it was biased towards the persons. I was more likely to get purple and Shohini green.

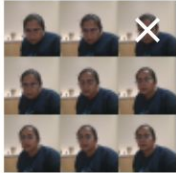
INPUT



Number of the top 20 closest matches in each class: 10,10,0  
Confidence for which the image matches each class: 0.5,0.5,0

OK

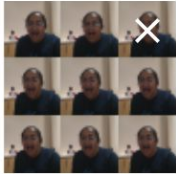
**133 EXAMPLES** **CONFIDENCE**



32%

**TRAIN GREEN**


**118 EXAMPLES** **CONFIDENCE**



67%

**TRAIN PURPLE**

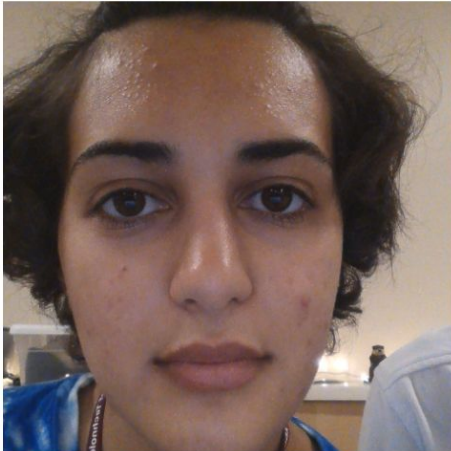
**0 EXAMPLES** **CONFIDENCE**



**Task 2-5:** Try a similar experiment, training class 1 as person A's face close up and class 2 as person B's face far away. Now move your faces closer/farther. How does the network behave? What happens to the network's performance and confidence if you add a third class with person B's face close up?

**Action 2-5:** Here it was very accurate! It was even able to recognize a friend's face though I do wonder if it was more recognizing the absence of the background.

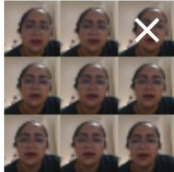
INPUT



Number of the top 20 closest matches in each class: 19,1,0  
Confidence for which the image matches each class: 0.95,0.05,0

OK

101 EXAMPLES

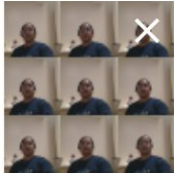


CONFIDENCE

97%

TRAIN GREEN

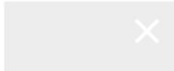
154 EXAMPLES



CONFIDENCE

TRAIN PURPLE

0 EXAMPLES



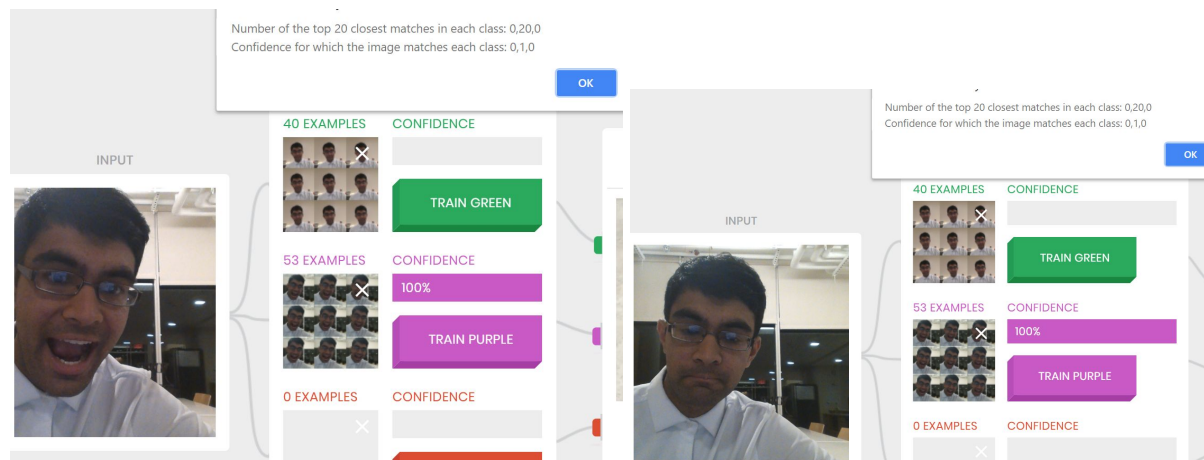
CONFIDENCE



**Task 2-6:** Read the Russian tank parable

([http://lesswrong.com/lw/7qz/machine\\_learning\\_and\\_unintended\\_consequences/](http://lesswrong.com/lw/7qz/machine_learning_and_unintended_consequences/)). Create some examples that illustrate this phenomenon. Some of the face examples above may already illustrate this, but make other examples.

**Action 2-6:** Here I trained the program on my facial expressions while changing the background and it indeed was overpowered by the setting change and only detected that.



**Task 3-1:** Look at the code in `computeConfidences()` and try to understand what it does. You can see that it computes the confidence for each class by just adding up the number of the TopK images in each class (the values of `nCounts` in the code) and dividing by K (the value of `kVal`).

Modify the code in `WebcamClassifier.js` to experiment with different ways of computing confidences, such as weighted vs. not weighted. You might want to keep things so that confidence values can be interpreted as probabilities, that is, so that each value is between 0 and 1 and so that the confidences for all three classes sum to 1.

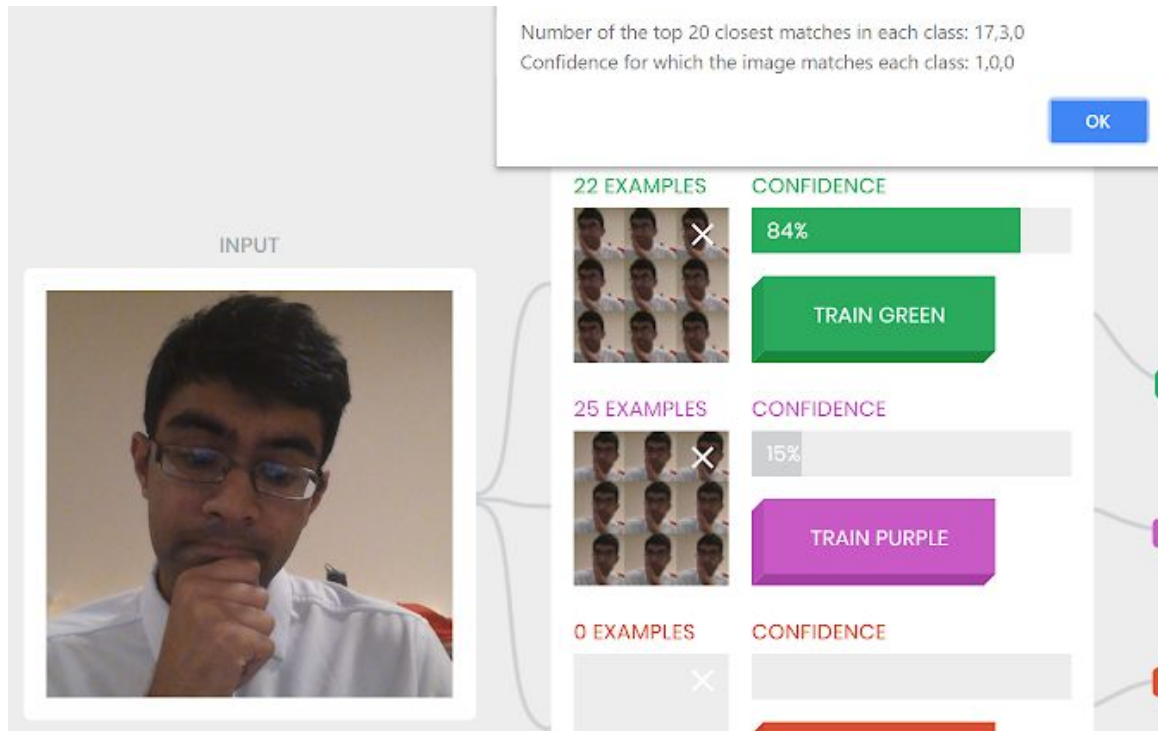
**Action 3-1:** There are many ways to adjust the confidence values. One method is to just assign 1 to the most likely match. I did this in the first version with the variables `maxConf` and `maxPool`. A second method I tried was noise reduction. Basically everything is treated the same as the default except that to be considered by the function you need to meet a lower threshold. In this case, I set it to 20% or 4 images.

```
let nCounts = [0, 0, 0];
let confidences = [];
let maxConf = [0,0]

for (let index = 0; index < CLASS_COUNT; index += 1) {
  nCounts[index] = classTopKMap[index];
  const probability = classTopKMap[index] / kVal;
  confidences[index] = probability;
  if (confidences[index] >= maxConf[0]) {
    maxConf[0] = confidences[index];
    maxConf[1] = index;
  }
}

let maxPool = [0,0,0];
maxPool[maxConf[1]] = 1
confidences = maxPool
```





```

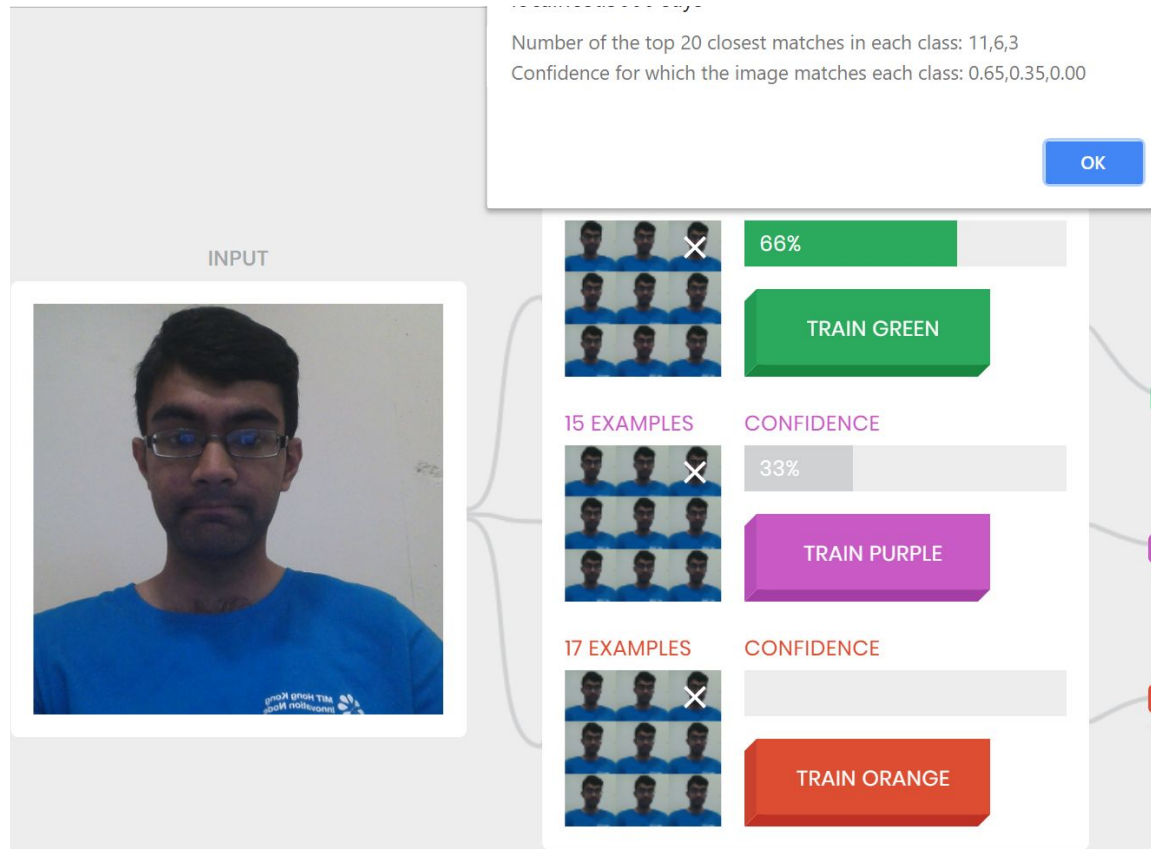
let nCounts = [0, 0, 0];
let confidences = [];
let evenConf = [0,0,0]
let totalConfvals = 0

for (let index = 0; index < CLASS_COUNT; index += 1) {
  nCounts[index] = classTopKMap[index];
  const probability = classTopKMap[index] / kVal;
  confidences[index] = probability;
  if (confidences[index] >= 0.2) {
    evenConf[index] = confidences[index];
    totalConfvals += confidences[index];
  }
}

for (let index = 0; index < CLASS_COUNT; index += 1) {
  evenConf[index] = (evenConf[index]/totalConfvals).toFixed(2);
}

confidences = evenConf

```



**Task 3-2:** Using your modified confidence calculations, try some of the example scenarios that confused the network, that you looked at in section 2 and section 4. Does one of your new methods perform better? Why/why not do you think so? Document and turn in the modified and unmodified results for a “confusing” example. Use screen capture to include images if appropriate.

**Action 3-2:** The maxConf and evenConf values are helpful in the same circumstances depending on your goal. For example, the images above show the results when the programs were trained on similar, but not exactly the same facial expressions. maxConf was helpful when you wanted to just show the strongest match and run with that, which may be helpful in making a strong decision. evenConf was helpful when you wanted to keep the probabilities of matches, but only the strongest ones - you wanted to exclude the most dissimilar matches but still have options to choose from.

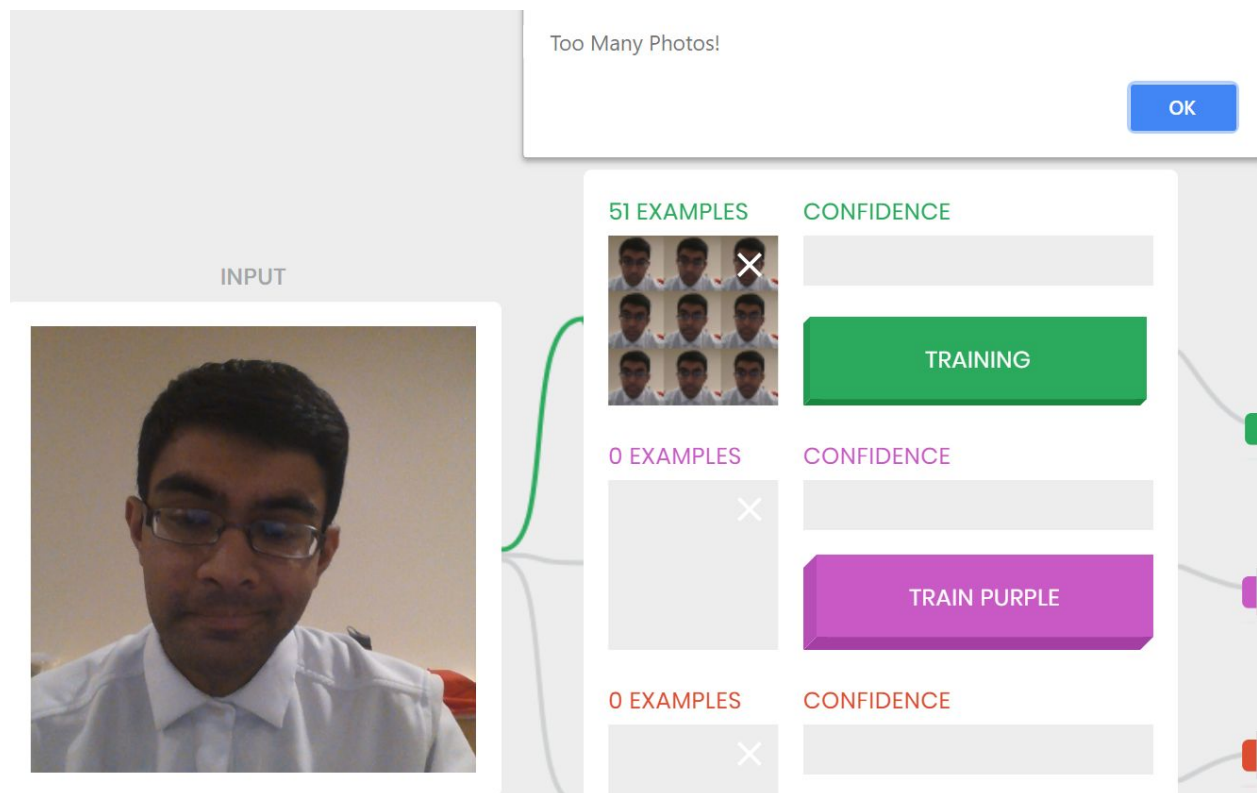
**Task 3-3:** What are some image classification situations where your alternative ways of confidences might come in useful?

**Action 3-3:** Both of these functions would be helpful when you have more than three options. As you get near 5+ possible matches (like green, purple) or choose to classify with more than 20 similarities you will get a lot of downstream noise and so these are options to deal with such circumstances.

**Task 4-1:** In this section, you will implement code to limit the number of training images that can be fed into each class. Then you will explore how the amount of training data affects recognition. Change *animate()* so that each class has an allotted number of samples it can accept. Once that number is reached, an alert window should pop up notifying the user and no more samples will be accepted for that class.

**Action 4-1:** I implemented the `window.alert` using the `if,throw` structure.

```
if (this.current.imagesCount > 50) {
  throw window.alert('Too Many Photos!')
}
```



**Task 4-2:** Once you've finished the implementation, explore what happens if the number of training images for the classes is severely unbalanced. Try using both easy-to-classify cases and some of the cases that confused your system previously.

**Action 4-2:** This was very interesting! Basically what happened was that if the training images were unbalanced there was a bias towards the classes with more data. In the extreme, if you did not have 20 examples you could never have a 100% match. If you had less than 10 examples, you would never even be in the majority!

You cannot see it in the image below but the green class was only given 9 examples!

Number of the top 20 closest matches in each class: 9,11,0  
Confidence for which the image matches each class: 0.45,0.55,0.00

OK

INPUT

48 EXAMPLES CONFIDENCE 45%

TRAIN GREEN

0 EXAMPLES CONFIDENCE 55%

TRAIN PURPLE

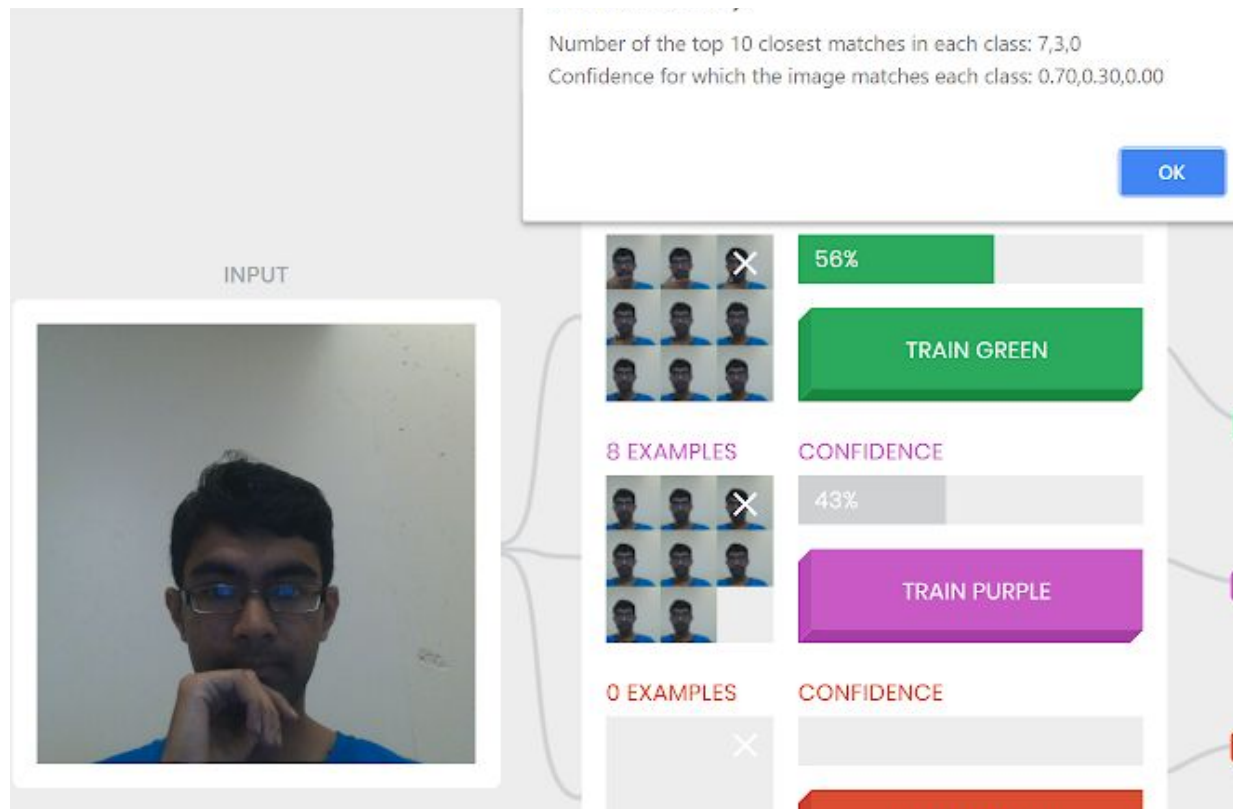
0 EXAMPLES CONFIDENCE

**Task 5:** Now that you have some familiarity with the code, explore some other interesting questions. Pick at least one of the ideas below. Feel free to do more and also explore any other interesting questions you think of. Write up your results to turn in. Your write-up should be a paragraph or two *at most* for each question. Please include your code and relevant images.

1. Adjust K and see how this affects accuracy

A higher K value is mixed blessing. As K increases and you are taking more photos to compare the test case with, you end up eliminating chance in matching. For example, in the extreme, where  $K = 1$  you would risk have a lot of false matches because a photo of you smiling just needs to be most similar to one of you frowning - perhaps even due to bad data or a boundary frown - and you would have a 100% match with frowning when if you have  $K=20$  it may only have been a 5 - 20% match. On the other hand, a high K value can be bad as well as you get more statistical noise. If you are trying to find 40 matches with three classes, each having 40 training data then there is a good chance you will have overlap with other classes even if its a decently solid match. If  $k > 40$ , you definitely would.

```
const IMAGE_SIZE = 227;
const INPUT_SIZE = 1000;
const TOPK = 10;
const CLASS_COUNT = 3;
```



2. How does having more/fewer classes affect how K relates to accuracy?

The more classes you have the higher you need K to be to prevent false matches overwhelming the probabilities. On the other hand you do increase the chance of false matches by choosing more wrong data points as well if you do not have enough data per class. All this to say that as

you have more classes you should increase  $K$  and can maintain a high accuracy but only if you have enough data to do so.