Pervasive Computing - Final Project

Ingredient recognition : Recipe Suggestions

Alexia - Maria Buliga, Diana - Elena Toderascu, Johnathan Razab-Sekh

Period 2 : fall / winter 2023

Vrije Universiteit Amsterdam

Table of contents

1. Introduction & Motivation

The goal of this project is to make a system that can suggest recipes to the user, based on one main ingredient of their choice . We took inspiration from the *Brickit* app that works as follows : when shown a multitude of LEGO pieces, it recognises each of them and informs the user what new creations could be made. Due to lack of time, we decided it would be a better choice to focus on just one ingredient at a time, as it would still be an useful thing to have in day to day life, helping people who are either indecisive about what to cook for their meals or who would like to use what they have in their homes. It would be especially helpful to students like us, who only recently started to live on their own and are still adjusting to cooking for themselves all the time.

2. Related Work : Literature Study

Even though we were optimistic about the feasibility of our idea from the beginning, we still took our time researching our area of interest. We checked to see if this had been done before, and if it had, to see how others approached the concept. Below you can find the two papers we found to be most relevant to our project, and our takes on them.

A Cooking Recipe Recommendation System with Visual Recognition of Food Ingredients - Keiji Yanai, Takuma Maruyama and Yoshiyuki Kawano, The University of Electro-Communications, Tokyo, Japan [1]

The first paper we came across during our research was from the University of Electro-Communications, Japan. In this article, a similar concept is discussed : a mobile app that can be used during daily trips to the grocery store, to aid people in finding ideas for their meals.

To implement their application, the developers also heavily relied on the classification of images and their different features, making this paper a great source of inspiration.

The article discusses taking colored pictures and turning them into 12 x 12 grids for the A.I. to use. We found that the 12 x 12 grids turned into really blurry outputs, so we decided to use 25 x 25 grids instead. Nonetheless, the grid technique was of great help to us in creating our project.

Besides this, we were also impressed by how much the writers were able to achieve working with only RGB colors. The article contains graphs that show just how little of a difference there is between the performance of the RGB method and those of other, seemingly more advanced options. This made us so much more confident in what we could do with our knowledge and our ideas together.

Assessing The Importance Of Colours For CNNs In Object Recognition - Aditya Singh, Alessandro Bay, Andrea Mirabile, Zebra AI, Zebra Technologies London, United Kingdom [2]

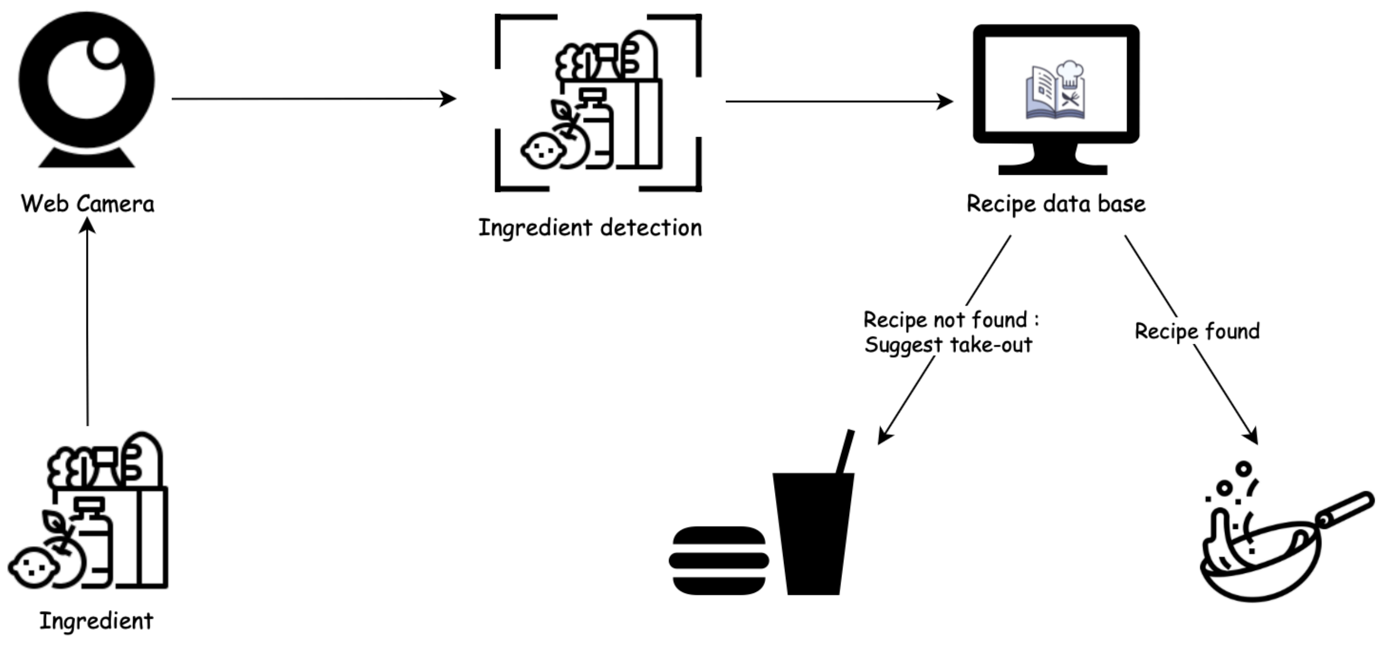
This paper took a little longer to find, but the effort and time were worth it.
Unfortunately, we couldn't use a pre-trained network like we originally intended, so we had to do it from scratch. A network that has been pre-trained is essentially different from one that was made from scratch, so we knew our approach had to be different. One of the problems we had was deciding if we want to work with coloured images or transform them to their grayscale versions.

Even if the research revolves more around CNNs, the writers have also discussed neural networks on a more general level. One of the main ideas that we got from reading it was that neural networks made from scratch are more dependent on color than their pre-trained counterparts.

With the new knowledge discovered we decided to not turn the images into grayscale and then store the information in a vector, even if it would have been easier. By choosing to work with coloured images we have assured a higher success rate than our original idea.

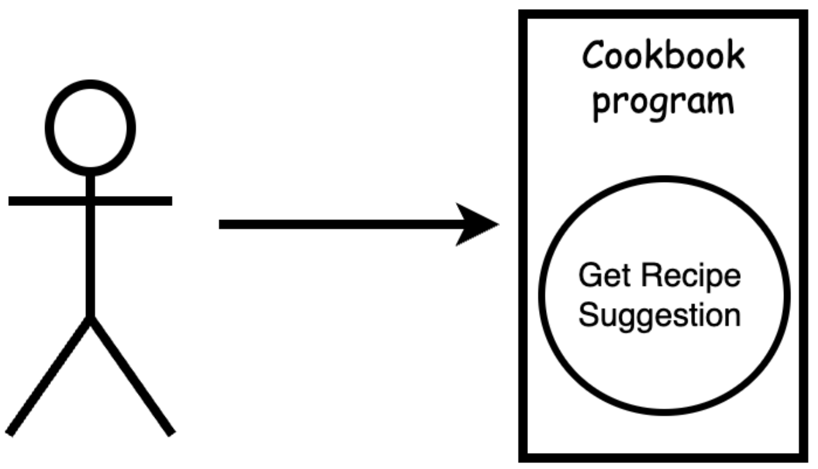
3. Concept of operations, system block diagram, user scenarios

Block diagram (system overview)



Use case diagram

Our use case diagram is rather simple since the user has only one way of interacting with the program : by showing it ingredients.



Stakeholders of this project :

- The developers of the project;
- The mentors who requestested the program;
- The final users who will use the program;

Different scenarios :

We believe that our system could be of help in a multitude of situations such as helping people when:

- They crave something specific and would like to make a whole meal centered around it;
- They are on a tighter budget and would prefer using what they already have in the house;
- They are simply lacking inspiration for their meals.

Advantages of the system :

Our program has the advantage that it can be expanded and made into so much more than it is now. For example, it can be customized to accommodate a lot of different user needs such as :

- Taste preferences;
- Dietary requirements;
- Medical requirements;

4. System requirements

When accessing this application, the user should be able to inquire about the potential use of an ingredient *(that our model has been trained on)* in one culinary recipe. The user has to take a photo of the item in question to be able to obtain the possible cooking method.

Functional Requirements

Must have:

- the ability to store a database of frames with the ingredients it was trained to identify ;
- take and store external photos for identification ;
- modify the photo by resizing it to 25×25 pixels ;
- classify the image as one of the classes created for the training process ;
- depending on which class it categorized the image to be in, display the appropriate recipe ;
- fairly intuitive user interface.

Can have:

- a bigger data-base to properly identify more ingredients;
- advanced data-augmentation to modify the training set;
- real-time image processing;
- better user interface.

Won't have:

- internal visualization of the neural network.

<u>Non-Functional Requirements</u>

Must have:

- the performance of approximately 15 seconds;
- a minimum of 75% accuracy in detecting the items.

Can have:

- a better performance;
- bigger accuracy in detecting items;
- error handling.

Won't have:

- the possibility for a user to have their own account.

<u>Constraints</u>

- not enough items to train the system with more items;
- the time before the deadline was ephemeral;
- MATLAB was used for this project.

5. System design

Introduction

Having done our research helped tremendously, since we knew that our idea was actually feasible within the given time and wouldn't require too much additional research while working. Of course we ran into problems along the way, but we successfully overcame these challenges with the help of the past labs and advice from our TA.
All of the code mentioned below is written in the category **"Appendix"** and is substantially better commented in the code files. Some of the lines mentioned in this document do not match the lines of code in the separated files.

Hardware

Our program is extremely unsolicitating on the computer. The only true requirement is having MATLAB installed to be able to run it properly and a webcam to be able to take a picture. Down there is a list of recommended hardware for smooth running created while taking into consideration MATLAB requirements.

- 8GB RAM
- Intel / AMDx86-64 processor with cores (Intel 7 – 1355U[1])
- 500GB SSD (500GB SSD)
- any modern GPU (Intel UHD Graphics[1])

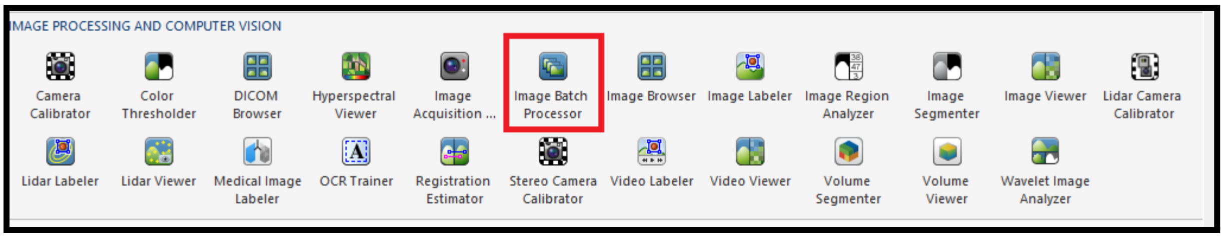
1 – these are the specifications for the most used laptop for the creation and testing of the system

Database Creation

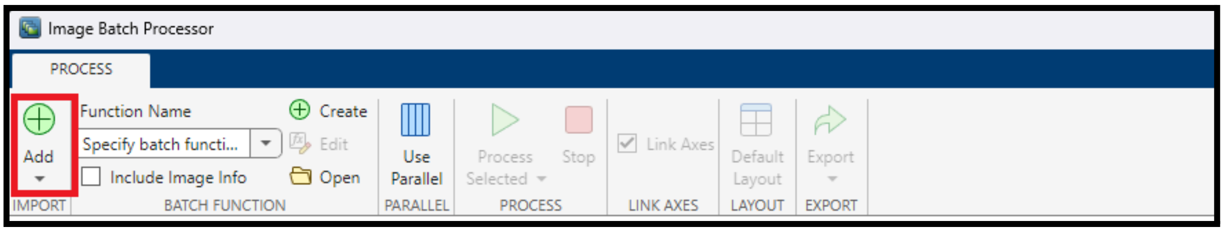
To generate the database necessary for the training and testing in a fast and efficient manner, we decided to film our items of interest in different positions and different lighting angles. Afterwards, we used a MATLAB code to extract every frame from the acquired videos.
The script has been taken from the MATLAB forum and we updated it to be able to work on the 2023 version. We will briefly explain it here.

extractFrames.m:
The code saves inside a variable *"vid"* a specified video that has to be inside the same folder. Another variable will take the number of frames using the command *"vid.NumFrames"*. The script ends with a *"for"* loop that iterates through all the frames inside the video and saves them under the title *"Image[i].jpg"*, the letter *"i"* representing the number of the frame.

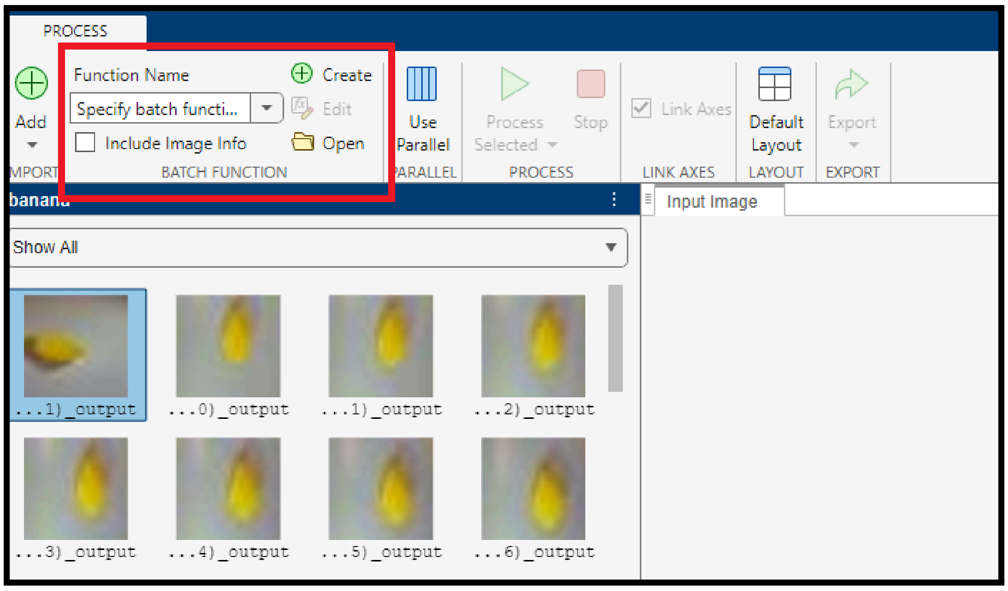
After extracting the frames we needed a way of processing a large amount of images into the 25x25 pixel format. We did this by using the "Image Batch Processor" app.



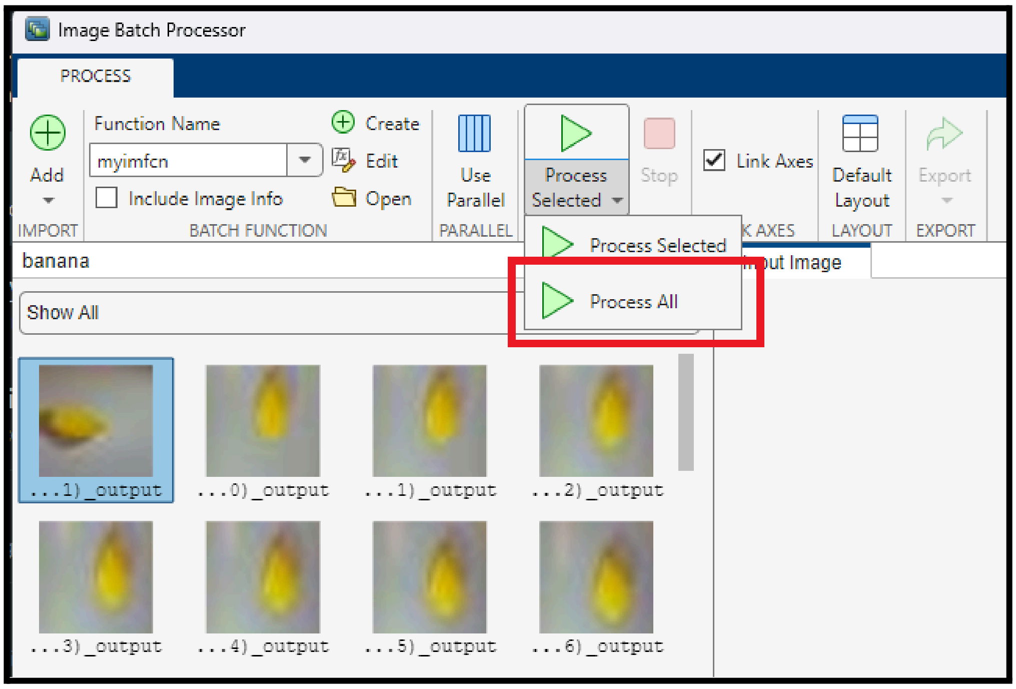
To use this app you first have to load the folder containing the images, this is done by pressing "Add" (see figure below).



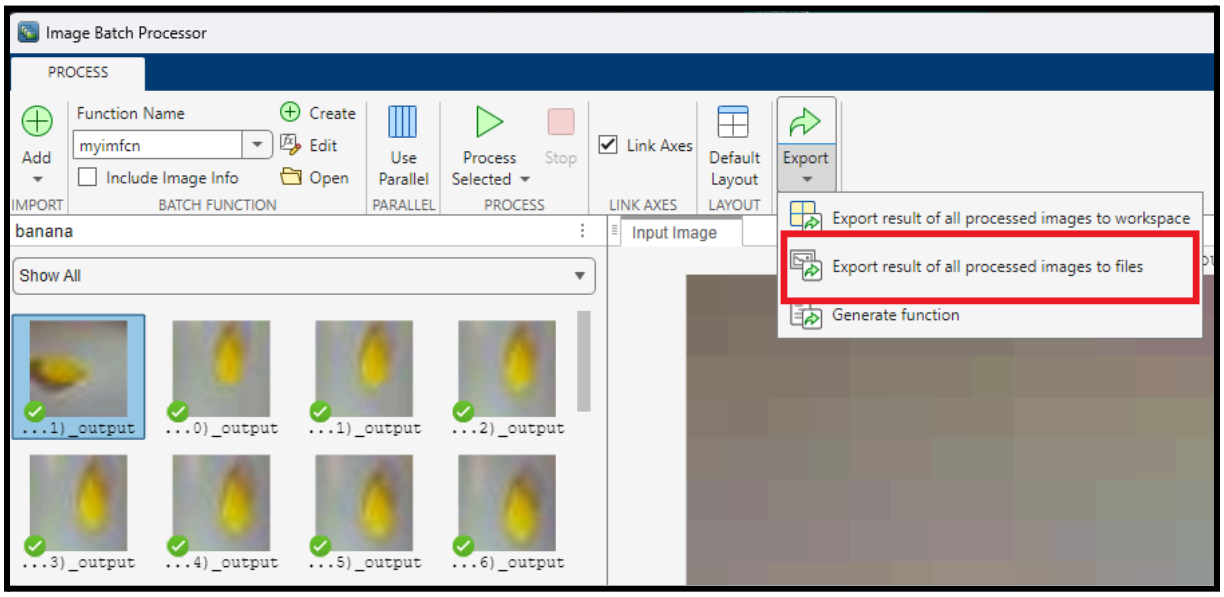
After loading the images you're able to see every image in the folder within the app, now you have to create or load a matlab file, our file is called *myimfcn.m* (that's the default name, it's what you get when you press"create" instead of opening an existing file).

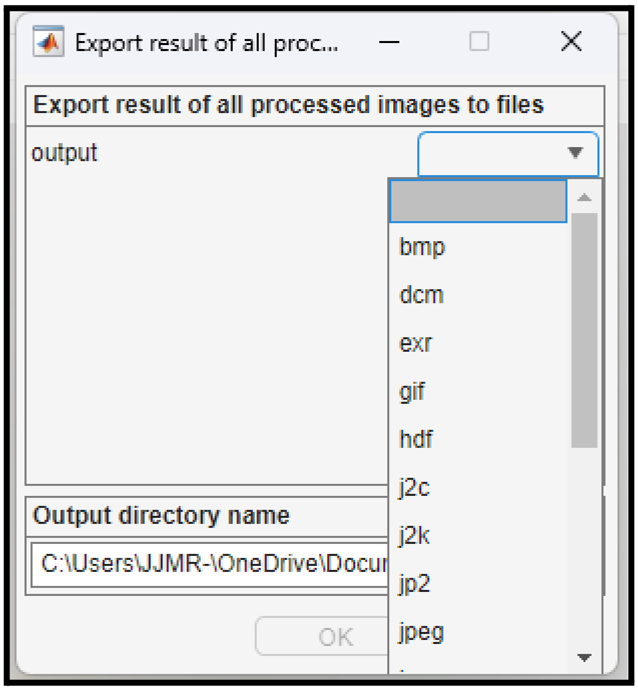


After creating/loading a file, you can press "Process All", this can take a bit (depending on how many images need to be processed.



When the Image Batch Processor is done processing all the images, you can press "Export result of all processed images to files". This will open a pop up menu where you're asked to select a file type (we chose jpeg) and a folder to save the images to.




Export result of all proc...
Export result of all processed images to files
output
bmp
dcm
exr
gif
hdf
j2c
j2k
jp2
jpeg
Output directory name
C:\Users\JJMR-\OneDrive\Docu
OK

We repeated this process for all of the classes, since we had our different data classes saved in separate folders. Because of the size of our dataset this process took some time to complete.

After saving the 25x25 images in a folder we use the code stored in the file *createMatrix.m* to store all the data of these images in a matrix.

Lines 1-3 show what label we attach to the different classes and how many images will be stored in the dataset.

Line 4-5 find the location of the folder containing the images and stores all the .jpeg files so they can be added to the matrix later.

Line 8-9 creates a matrix of zeros, the amount of rows is based on how many files there are but the amount of columns will always be 25x25x3+1 (25x25 because of the image size x3 because it is an RGB image and +1 to make space for the label). The name was either "turkey_matrix", "banana_matrix" or "chicken_matrix" depending on which folder of images we were working with.

The last lines are used for the for loop, this will iterate through the images in the folder and read it into a variable "im", next it will create a variable "im_vector" which stores a reshaped version of the "im" variable with the dimensions 1x(25x25x3). The i-th row and first column of the big vector will next be appended to the label (what label it is supposed to contain is shown in the first 3 lines). Finally, starting at the 2nd column of the big matrix all the way to the end will contain the im_vector. since it's a for loop, this repeats for each image in the folder.
We ran this code 3 times for the 3 different classes and then used the *createAndShuffle.m* file.

With this big data set we could finally ready this dataset for the intended purpose, training our network. This was done with the file *prepForNN.m*. These lines of code match what we were taught during lab 4. We also used what we learned during lab 4 to create and train the neural network.

Neural Network

When we first started working on this project, after much documentation and research, we decided to use a pre-trained NN through the Deep Network Designer app offered by MATLAB. More precisely, we started with ResNet50. We decided to use a pre-trained one so we can have better accuracy when identifying our ingredients.

Unfortunately, this strategy quickly failed after encountering an error that would have taken more time to resolve than starting fresh with a new approach.

We decided to start from scratch with a “fresh” neural network. We used our modified frames to train it to identify three items: a chicken, a turkey leg and a cluster of bananas. The frames were separated into three categories:

- 70% - Training Data
- 15% - Testing Data
- 15% - Validation Data

Additionally, we kept the default layer size of 10, as this seemed to work best.

Final Results

With our neural network fully trained and prepared, we have written a code that takes input, gives it to the network, reads the output and then prints the recipe assigned for the specific ingredient.

classifyImage.m:

The first segment (lines 2 - 7) takes a video of one second with the camera attached to the device. It will set the color space of the video to RGB in order to optimize the performance of the neural network in detecting the ingredient. Afterwards, we will store one frame from the video in a variable *"im"* and delete the video object to ensure no leaking will take place.

Line 10 resizes the frame to a 25×25 pixel size so that the network will be fully able to classify the ingredient in one of the categories.

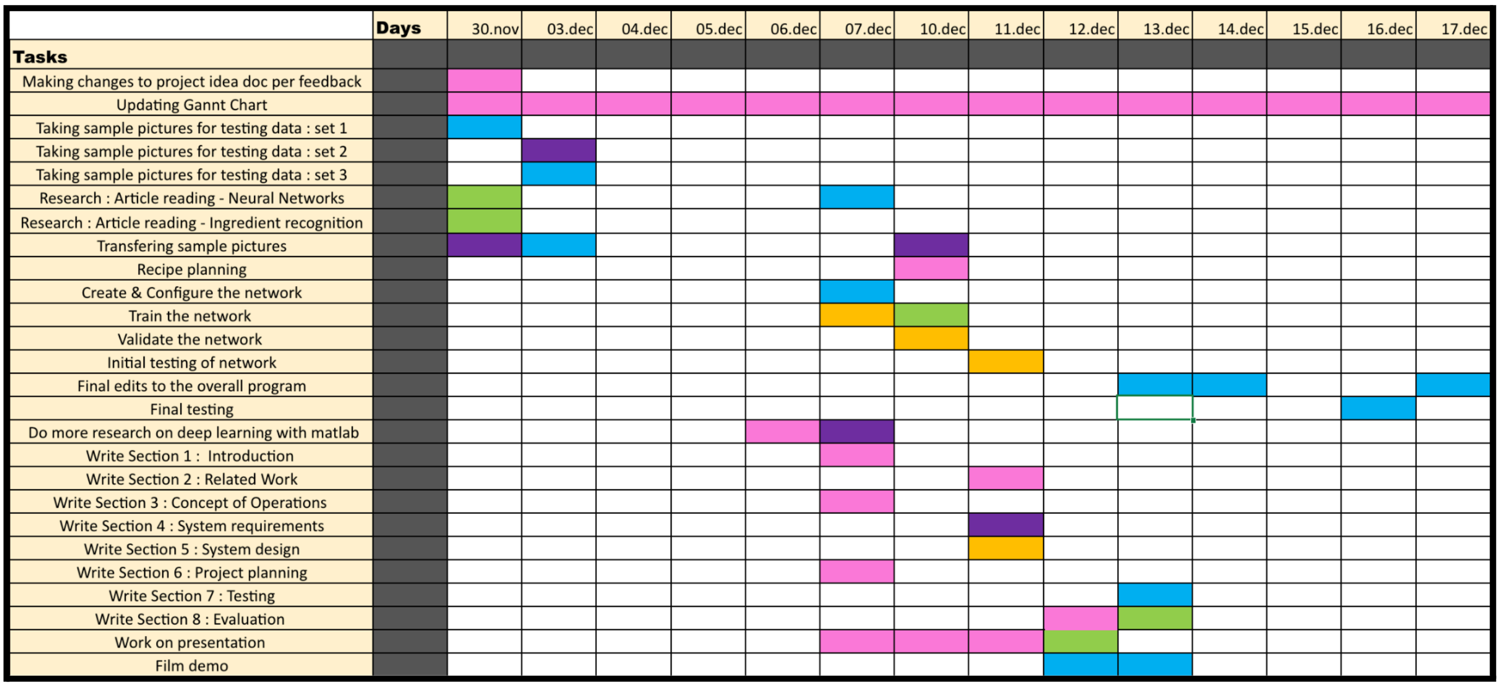
Lines 13 - 16 store the data of the image into one row and then add it into a matrix so the neural network can take it as input and classify it.

After running the image through the network, the script reads the final result. It will plot the original image and the modified one that was used in detecting the item. Using an *"if-else"* statement it will display, as a title to the second image, the name of the recipe for the product detected.

6. Project planning

Task division & Timeline

As requested, we created a Gannt chart to keep track of our tasks and time. In the chart, everyone has their own color, and shared tasks are marked appropriately.



	Days	30.nov	03.dec	04.dec	05.dec	06.dec	07.dec	10.dec	11.dec	12.dec	13.dec	14.dec	15.dec	16.dec	17.dec
Tasks															
Making changes to project idea doc per feedback		🟥													
Updating Gannt Chart		🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Taking sample pictures for testing data : set 1		🟦													
Taking sample pictures for testing data : set 2			🟪												
Taking sample pictures for testing data : set 3			🟦												
Research : Article reading - Neural Networks		🟩					🟦								
Research : Article reading - Ingredient recognition		🟩													
Transfering sample pictures		🟪	🟦					🟪							
Recipe planning								🟥							
Create & Configure the network							🟦								
Train the network							🟧	🟩							
Validate the network								🟧							
Initial testing of network									🟧						
Final edits to the overall program											🟦	🟦			🟦
Final testing														🟦	
Do more research on deep learning with matlab						🟥	🟪								
Write Section 1 : Introduction							🟥								
Write Section 2 : Related Work									🟥						
Write Section 3 : Concept of Operations							🟥								
Write Section 4 : System requirements									🟪						
Write Section 5 : System design									🟧						
Write Section 6 : Project planning							🟥								
Write Section 7 : Testing											🟦				
Write Section 8 : Evaluation										🟥	🟩				
Work on presentation							🟥	🟥	🟥	🟩					
Film demo										🟦	🟦				

Elena's tasks : purple
Maria's tasks : pink
John's tasks : blue
Shared tasks between Elena and John : orange
Shared tasks between the three of us : green

Our strategy was to plan our tasks in such a way that we could make the most out of each of our individual abilities while still leaving room for us all to learn new things. This worked out really well for us, making the whole project run a lot smoother.

Risk analysis

- The files that contain both code and training data are stored on our personal computers and not on a drive/ cloud so there is the risk of the computer crashing and us losing the files. To combat this issue, we will share the files as equally as possible so in case the worst happens we will not lose all progress. Additionally, we will create the main document, presentation and Gannt chart online, so that they will be safe at all times ;
- We originally intended to train the program to recognise multiple ingredients at once, but for the sake of time, we decided not to take the risk. Our plan is that if at the end there will be any remaining time, we will expand the project ;
- We will leave the recipe planning for the later part of the project because we think there is no point in doing that in the beginning, just in case the actual program takes longer than expected and we end up not using them and losing time.

7. Testing

For testing purposes we made a separate file called *classifyFromMatrix.m* this file was used purely for testing purposes. To perform our tests we created 3 matrices filled with images of one of the 3 classes (one matrix called “turkey_matrix”, one called “banana_matrix” and one called “chicken_matrix”).

Line 1 copies one of the 3 matrices over to a matrixName variable (this made it easier to test the 3 different matrices since we only had to change one variable)

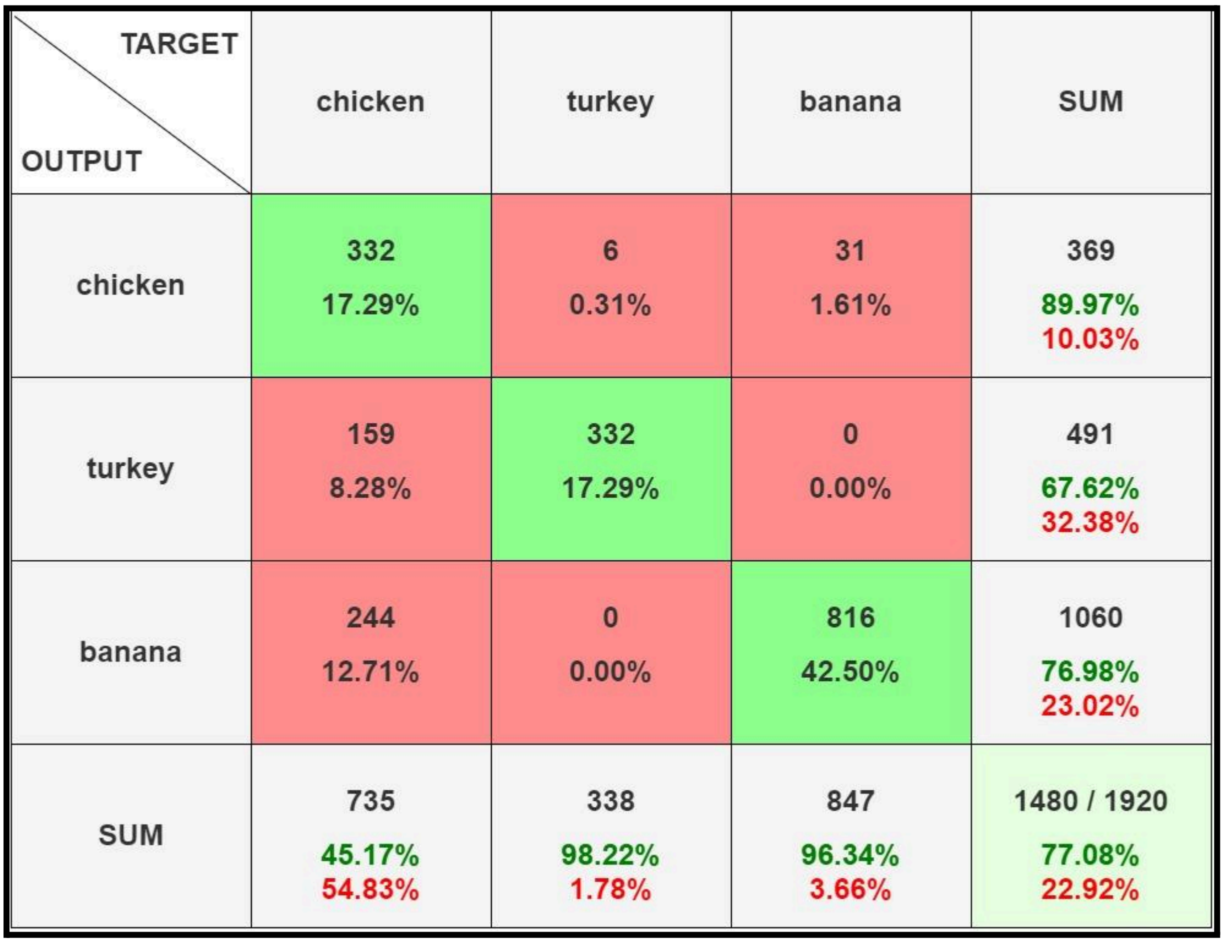
Line 3-4 are used to create a matrix used to store all the output of the network

line 7-14 contains a for loop that iterates through the input matrix and runs every row through the network. After this it reads the output from the network and stores the index of the highest value in the testNetworkOutput variable.

Line 17 finds all the unique values in the testNetworkOutput
line 20 counts all the occurrences of the different unique values

Line 23-26 is another for loop that, for every unique value found in the testNetworkOutput matrix, prints how many times it occurred.

We used this code to gather the data necessary for the following confusion matrix:



TARGET / OUTPUT	chicken	turkey	banana	SUM
chicken	332 17.29%	6 0.31%	31 1.61%	369 89.97% 10.03%
turkey	159 8.28%	332 17.29%	0 0.00%	491 67.62% 32.38%
banana	244 12.71%	0 0.00%	816 42.50%	1060 76.98% 23.02%
SUM	735 45.17% 54.83%	338 98.22% 1.78%	847 96.34% 3.66%	1480 / 1920 77.08% 22.92%

In this confusion matrix we see that the network has a success rate of 87.95%. We are very satisfied with this result, especially considering our lab schedule was not ideal and Johnathan was also sick for a week and unable to make much progress during this time.

We see that the chicken gets classified perfectly, which is obviously what we strived for during all 3 classes while creating the network, but the turkey is misclassified a lot of the time. This is most likely due to the similar color to the chicken. Surprisingly the banana has a similar problem where it gets classified as a chicken.

On closer inspection we noticed that the banana’s misclassification is most likely caused by lighting. we saw that when the room is a little dark the color of the banana looks more brown-ish and closer to the plastic chicken. This coupled with the fact that a 25x25 pixel format lost a lot of shape detail resulted in misclassification.
The same could be argued for the turkey, it gets misclassified because of lighting. The colors of the chicken and turkey are fairly similar and in certain lightings they become even more similar, coupled with the lack of clarity on the shape of the object results in misclassification.

8. Evaluation

Overall, this project was a great opportunity for us to dive more into the broad world of A.I. . Even though we went through some trials and errors during our attempts to make the program work, we still found the tasks to be quite interesting.
As for our teamwork, now that the project is done, we can say that all the members of the group are satisfied with the way we approached things. We kept in touch through different platforms such as Discord and Whatsapp, making sure to update the others about whatever progress was made each day. As we mentioned in the planning section, we tried our best to split tasks fairly between the 3 of us, so that no one would end up being burnt out from the workload. So as far as the actual teamwork goes, we are happy with the way we handled things.

At the beginning we might have been a bit overly-ambitious, trying to work with multiple food items at the time. However, we quickly switched up our approach after evaluating the risk of not finishing in time, and decided it was not worth it.
If we were to do this again, there are a few things that we would do differently :

- we would try to gather even better training data so that our program could have a better understanding of the things it needs to recognize;
- we would check more than once what we are allowed to do and what we are not, since a little bit of time was lost at the beginning because we tried to use a method that was not presented in the course and therefore, not really allowed.

  Other than this, all 3 members are quite satisfied with their contribution to this assignment and the group work overall.

9. Appendix

extractFrames.m [3]

```
vid=VideoReader('video.mp4');
numFrames = vid.NumFrames;
n=numFrames;
for i = 1:2:n
    frames = read(vid,i);
    imwrite(frames,['Image' int2str(i), '.jpg']);
    im(i)=image(frames);
end
```

createMatrix.m

```
%chicken(1) = 4763
%turkey(2) = 9858
%banana(3) = 3277
image_folder =
'C:\Users\JJMR-\OneDrive\Documenten\MATLAB\project\images\turkey';
files = dir(fullfile(image_folder, '*.jpeg'));
% Initialize your dataset matrix
num_images = numel(files); % Number of images you have
turkey_matrix = zeros(num_images, 25*25*3 + 1); % 25x25 image in RGB
(3) +1 for the class column
% For each image:
for i = 1:num_images
    % Read the image
    im = imread(fullfile(image_folder, files(i).name));
    % Convert the image to a row vector
    im_vector = reshape(im, [1, 25*25*3]);

    % Assign the class (0 for the first class)
    turkey_matrix(i, 1) = 2;

    % Assign the image pixel values to the dataset
    turkey_matrix(i, 2:end) = im_vector;
end
```

classifyImage.m

```
%take a photo
vid = videoinput ("winvideo",1);
vid.ReturnedColorspace = 'rgb';
start (vid);
im=getsnapshot(vid);
stop(vid);
delete(vid);
%make the image into a 25x25 pixel image
lowResIm = imresize(im, [25, 25]);
%prep image for network by storing all the data in 1 row instead of a
```

```
%25x25x3 matrix
im_vector = reshape(lowResIm, [1, 25*25*3]);
input_matrix = zeros(1, 25*25*3);
input_matrix(1, 1:end) = im_vector;
input_matrix = input_matrix';
%run it through the network
output = net (input_matrix);
%read output to find what class it belongs to. this is done by getting the
%index of the highest in the output matrix and from there we can find out
%what class the image belongs to.
[maxval, indexMax] = max(output);
subplot(2,1,1);
imshow(im);
subplot(2,1,2);
imshow(lowResIm);
if (indexMax == 1)
   title("Garlic Herb Butter Roast Chicken");
elseif (indexMax == 2)
   title("Apple Cider-Braised Turkey Drumsticks");
elseif (indexMax == 3)
   title("Banana Split");
else
   title("default output, shouldn't happen");
end
```

classifyFromMatrix.m

```
matrixName = chicken_matrix;
%matrix with dimension outputx1 used for all the output from the network:
numRows = size(matrixName, 1);
testNetworkOutput = zeros(1, numRows);
%iterate through input matrix
for i = 1:size(matrixName)
   test_input = matrixName(i, 2:end); % take the i-th image in the training set
   test_input = test_input'; % transpose it
   test_output = net (test_input); % run input through the network and store output in output variable
   [maxval, indexMax] = max(test_output); %read the output
   testNetworkOutput(1, i) = indexMax; %add output to output matrix
end
% Find all unique values in the output matrix
uniqueValues = unique(testNetworkOutput);
% Count the occurrence of every unique value
occurrences = histcounts(testNetworkOutput(:), [uniqueValues, max(uniqueValues)+1]);
% Displaying the counts
for i = 1:numel(uniqueValues)
```

```matlab
  fprintf('Value %d occurs %d times.\n', uniqueValues(i), occurrences(i));
end
```

createAndShuffleSet.m

```matlab
big_matrix = [turkey_matrix; chicken_matrix; banana_matrix];
num_rows = size(combined_matrix, 1);
shuffled_indices = randperm(num_rows);
shuffled_matrix = combined_matrix(shuffled_indices, :);
```

prepForNN.m

```matlab
labels=shuffledMatrix(:,1); % this is the first column containing the labels
labels(labels==0)=3;
targets=dummyvar(labels); %make the matrix targets
inputs = shuffledMatrix (:,2:end); % contains all the image pixels
inputs = inputs'; % transpose input matrix
targets = targets'; % transpose the target matrix
```

myimfcn.m

```matlab
function results = myimfcn(im)
%Image Processing Function
%
% IM      - Input image.
% RESULTS - A scalar structure with the processing results.
%
%--------------------------------------------------------------------------
% Auto-generated by imageBatchProcessor App.
%
% When used by the App, this function will be called for every input image
% file automatically. IM contains the input image as a matrix. RESULTS is a
% scalar structure containing the results of this processing function.
%
%--------------------------------------------------------------------------
% Replace the sample below with your code----------------------------------
results = imresize(im, [25, 25]);
%--------------------------------------------------------------------------
```

10. Bibliography

- A Cooking Recipe Recommendation System with Visual Recognition of Food Ingredients - Keiji Yanai, Takuma Maruyama and Yoshiyuki Kawano The University of Electro-Communications, Tokyo, Japan [1]
- Assessing The Importance Of Colours For CNNs In Object Recognition - Aditya Singh, Alessandro Bay, Andrea Mirabile, Zebra AI, Zebra Technologies London, United Kingdom [2]
- https://nl.mathworks.com/matlabcentral/answers/58111-how-to-convert-video-in-to-image-frames-using-matlab [3]
- https://nl.mathworks.com/matlabcentral/answers/495621-looping-each-pixel-of-matrix-image
- https://nl.mathworks.com/matlabcentral/answers/399613-how-to-lower-resolution-of-a-picture
- https://nl.mathworks.com/help/images/batch-processing-using-the-image-batch-processor-app.html#ProcessFolderOfImagesUsingTheImageBatchProcessorAppeExample-3
- https://nl.mathworks.com/help/images/batch-processing-using-the-image-batch-processor-app.html
- https://nl.mathworks.com/matlabcentral/answers/401480-how-do-i-run-all-images-in-a-folder-through-an-image-processing-code
- https://www.damianoperri.it/public/confusionMatrix/?noc=3
- https://matlab.fandom.com/wiki/FAQ#How_can_I_process_a_sequence_of_files.3F
- https://nl.mathworks.com/matlabcentral/answers/401480-how-do-i-run-all-images-in-a-folder-through-an-image-processing-code