Project scope

The goal of this project was to create an app which can recognize hand gesture from a user's camera or webcam. The user should just be able to show a gesture in front of their device and the app should give them a label for the gesture. The app needs to be reliable, which means that something like bad lighting or distance from the camera should not have a bit impact on the results. I also set some requirements that the app should comply with:

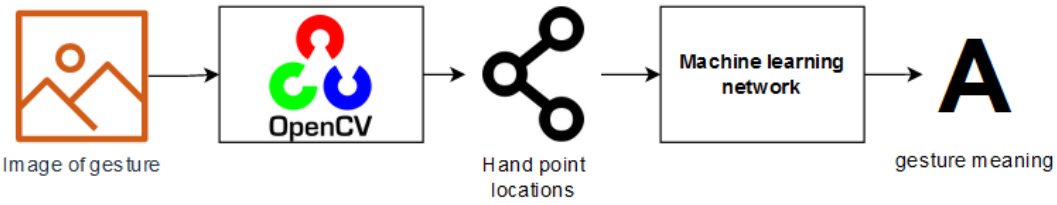
- The app should be able to recognize hands in real time.
- Both the left and right hand can be used to make a gesture.
- A debug view should be available where the user can see the recognized points, confidence and camera info like resolution and fps.

Furthermore, I decided that the entire project should be open source. Meaning that anyone can not only use the app. But also make tweaks to the processing, pre-processing, modelling. And use their own dataset. This does require good documentation to ensure that anyone can understand the thought processes.

Finally, with the time budget of around two weeks I had to also set constraints for myself. For example, I decided to only use a dataset with 10 labels. The dataset I chose to use contained images of all American sign language digits from '0' to '9'. The dataset only contained around 2000 images. This size allowed me to process everything quickly. Saving me time over using a much larger dataset.

Approach for model

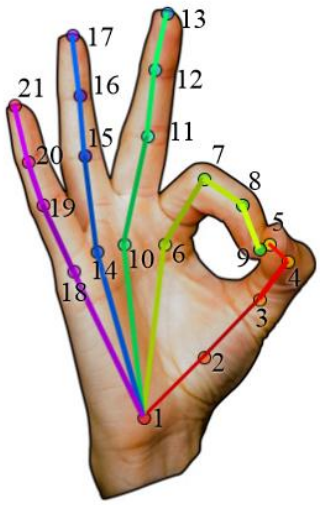
First, I started thinking about how I wanted to algorithm to work. I already knew that I wanted to use another algorithm to recognize the hand pose, and from these points I could create an algorithm to recognize the gestures.



Different models were used for processing the images and for in the app. For processing the images, I used OpenCV with an OpenPose model. OpenCV is an open-source computer vision library can use models like OpenPose to recognize human poses. It's a very robust model which can recognize hands in low resolution images. Which makes it a great candidate for processing a dataset with low resolution images. One downside is that it's slow. Recognizing a hand takes around a full second. So, it would not make a great fit for using in a real time app.

For the app I decided to use the Mediapipe-hands model. This is another open-source model which can recognize hands in real time. Making it a perfect fit for the app. It can also recognize multiple hands out of the box. Making it a great fit for the app.

Most hand pose estimation algorithms return a total of 21 points. These points are on the joints of the hand. Both the OpenPose model and the Mediapipe-hands model return these points. But the do it in different ways. The OpenPose model returns the locations of the points in pixel values. And the Mediapipe-hands model returns the location of the points in a value between 0 and 1. But after normalizing the values they come out as the same. This does mean that our model is able to accept many different hand pose estimators which return these 21 points.



METHOD	CARD	DESCRIPTION
Library	Available product analysis	First I looked online for available hand pose estimation models.
Field	Task analysis	I found out how most of the hand pose models function. And the values they return.
Showroom	Benchmark test	I tested the models to see which one would be better for certain use cases.

Approach for data

The dataset to be used needs to be put in the 'images' folder within the root directory. There are some limitations of the OpenPose model that need to be accounted for in the dataset. In the 'Dataset restrictions' notebook I go over a few. The exposure of the images cannot be too bright, I noticed that darker images preform better. Also, the resolution of the hand within the images needs te be greater than 60 by 60 pixels.

But this also depends on the threshold of probability you choose. It's reccomended that this threshold is between 0.1 and 0.25. If you go any lower the results will not be great because the almost all points are above the theshold. Even the bad ones. If you go too high it will be difficult to get many points because some of the points will be below the theshold and will be ignored. So you will have to find a good balance.

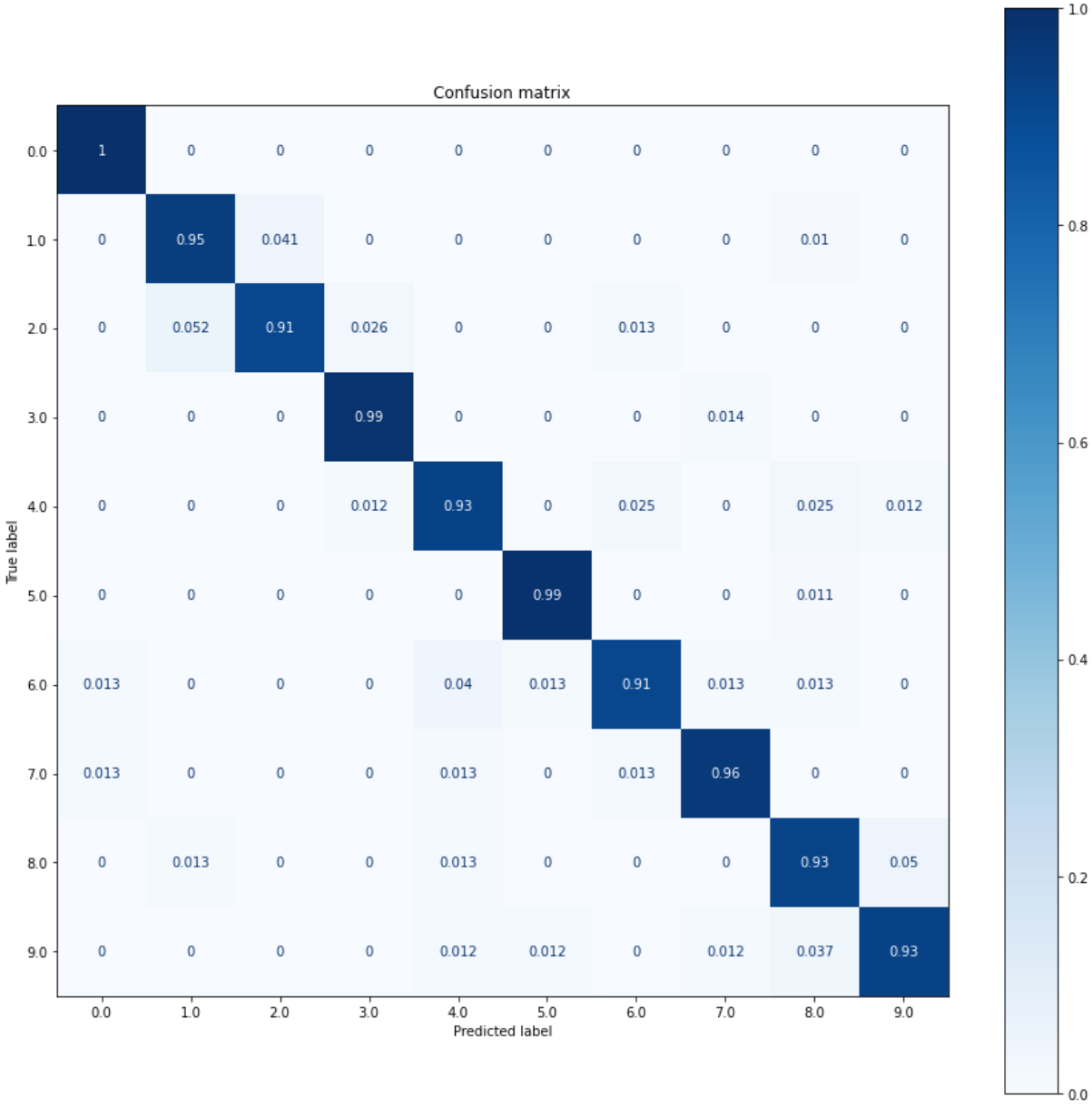
Furthermore, it might also not be smart to choose a datase which is too big. First this will take a very long time to process. And you might run the risk of over fitting the data. In the dataset which I used there are about 200 images per gesture. And I would also not go too low. Too little data will make the algorithm very inaccurate. I would recommend that go do not go below a 100 images per label.

Both the OpenPose model and Mediapipe-hands accept most common image files like: ‘jpg’, ‘jpeg’ and ‘png’. If you choose to create your own dataset I would recommend storing the files as either ‘jpg’ or ‘jpeg’, because they have a smallest file sizes of common image types.

METHOD	CARD	DESCRIPTION
Field 	Domain modeling	First the domain was found. I looked at the stakeholders: Deaf people, People who want to learn sign language & businesses.
Lab 	Data analytics	Then data was found which fit the criteria. Images containing hand gestures. Then example data was put through the OpenPose model and visualized.
Lab 	Component test	The limits of the Openpose were found to see if the data is usable with the model.

Approach for training model

For training the model I choose to follow the Sklearn cheat sheet. I already knew that I needed a classifier algorithm. I ended up on the LinearSVM algorithm. But after modelling this ended up not being the right one. I think it might have something to do with the flipped gestures. Because these are just the same gestures but flipped, I think it's difficult to draw a straight line through these areas. So, I decided to go with the poly kernel for the SVM algorithm. This gave me great results with a score of 0.95. Testing it with the test images, the results were great. It predicted all the gesture correct and the confidence was never below 0.8. For training I would recommend splitting the dataset into a test and train set to create both a score and a confusion matrix. I choose to split it 20/80. Where 80% is the training set and 20% is the testing set. The confusion matrix also gives us great results. All the labels were predicted correctly and there are no labels with a confidence below 0.9.



METHOD	CARD	DESCRIPTION
Library 	Design partern research	First I followed the Sklearn cheat sheet to find a suitable model for the dataset.
Lab 	Component test	Testing the model with the test set to validate and review the preformance.
Lab 	Unit test	Test the model with different types of image to validate every label/type of data.
Showroom 	Pitch	Wrote a conclusion for the model to briefly explain how it works and why it's better that is decoupled from the hand pose estimation model.