# **Highlights during development**

## Backend exploration and Data augmentation

* Most of the time during development was spent in understanding how convolution happens in deep learning models and how UNet model works. It was of key importance to understand how data is passed from downsampling path to upsampling path.
* Major difficulty in the development process was the absence of a backend, so a lot of reading had to be done to understand various semantic segmentation models. Many datasets are available online but datasets containing classIDs labels, classID based colormap labelled images and original images containing multiple classes together were hard to find. In the end VOC dataset was chosen since it contained colormap based labels and original images containing multiple objects.
* Understanding the workflow of convolution and filter creation, how features are learnt and which part of the models are important for classification.
* After a couple of brainstorming sessions, the goal of pixel based visualisation became more clear. Deconvolution using transposed convolution was the next step to understand upsampling and how pixels are classified during upsampling.
* Since GradCAM doesn’t depend on the backend, it’s structure could be analysed and applied to any semantic segmentation model. GradCAM visualisation can be stripped down to “Mapping colors to weights of pixels according to their importance w.r.t to class of the image”. In this project, GradCAM was applied to each pixel rather than the whole image.
* Again due to the absence of a concrete backend, data has been augmented to mimic a GradCAM data structure for each pixel in an image. For this a window of radius 50pixels around the pixel in focus is considered and upto 250 pixels are selected by generating them *normally* (gaussian distribution) so that density of pixels close to the principal pixel is higher. Random weights are generated for these chosen pixels.
* Pixel locations are then converted from 2D to 1D. Array of weights generated for each of these pixels is appended to create a 2xN matrix. 2xN matrix for each pixel in the image is generated and stored in a csv file.
* Odd numbered rows contain 1D pixel locations and even numbered rows contain weights for the corresponding pixel locations. Rows *n* and *n+1* are pixels and weights for GradCAM pixels of principal pixel *n*.

## Visualisation

* A lot of pixel data is generated since a lot of pixels are responsible for deciding the class of each pixel in an image. This leads to the csv file being really big (around 300-500mb). A library called Papaparse is used to read the csv file quickly (within 7-12seconds).
* One interesting observation while using the tool to prevent caching was to append a random number to the data path of *interesting.json* data. This makes sure whenever data is saved to json file, cached copy is never used.
* Care about UI is taken as to attract attention towards areas which are important. For example, the first thing which pops out when the tool is loaded are the interesting points, which are blinking circles implying they are interactive. Other areas have muted colors so that the users are not distracted by flashy elements which do not offer much interaction.
* Interactions with the tool are kept as natural as possible. Actions which most people are used to such as double click, single click, scrolling, Ctrl + Click, pressing S to save, etc are used.
* In canvas, pixel data is stored in a 1D map as opposed to 2D matrix. This is the reason why pixel positions were converted to 1D position while augmenting data. When a pixel is hovered, 2D pixel position is converted to 1D pixel position. Then this index is used to fetch appropriate rows from the csv data.
* Classlist containing class ID and it’s corresponding color is contained in a JSON format which is queried for each pixel in the image.
* Semantic segmented image and Original image are placed on each other. Visibility of either is controlled by scrolling on it. This gives the user freedom to focus on segmented or original images. This feature of switching between 2 images using transparency can also be employed to compare Ground truth labelled image and resultant image obtained from UNet etc models.
* Patch window which is created while hovering over the main image changes its width and height in real time according to the bounding box created by analysing pixel positions row.
* Upon clicking, a patch width of 160px is created. Pixels in the patch window are accordingly scaled up or down to fit the width constraint. This patch has an overlay containing GradCAM pixels colored using a colormap based on the weights row for the principal pixel.
* A slider filters out the pixels based on weights and helps the user focus only on a subset of pixels or all of them. Opacity of GradCAM pixels can be controlled upon scrolling so that the focus can be either on the patch or GradCAM or both.
* Patches are grouped according to its principal pixel’s class ID. Upon hovering these patches, context is preserved. Principal pixel in the main image is connected to this patch by a line and the bounding box is also created around the pixel.
* When a user marks a patch as interesting, the principal pixel is set interesting by a blinking circle. Even when the main image is changed, pixels marked interesting are preserved in a json array i.e. when the old image is opened again the interesting pixels are still there. To make these interesting pixels persistent, they can be saved into a json file. Visibility of these interesting points can be toggled since they can obstruct view at certain locations.
* A structure is recommended to store images and csv data in a folder so that it becomes convenient for the user to open images if interesting points are marked. Auto load is employed to do this i.e. original image, semantic segmented image and pixel data is loaded on a button press (by selecting image name button under *Interesting cases* without requiring much user interaction.