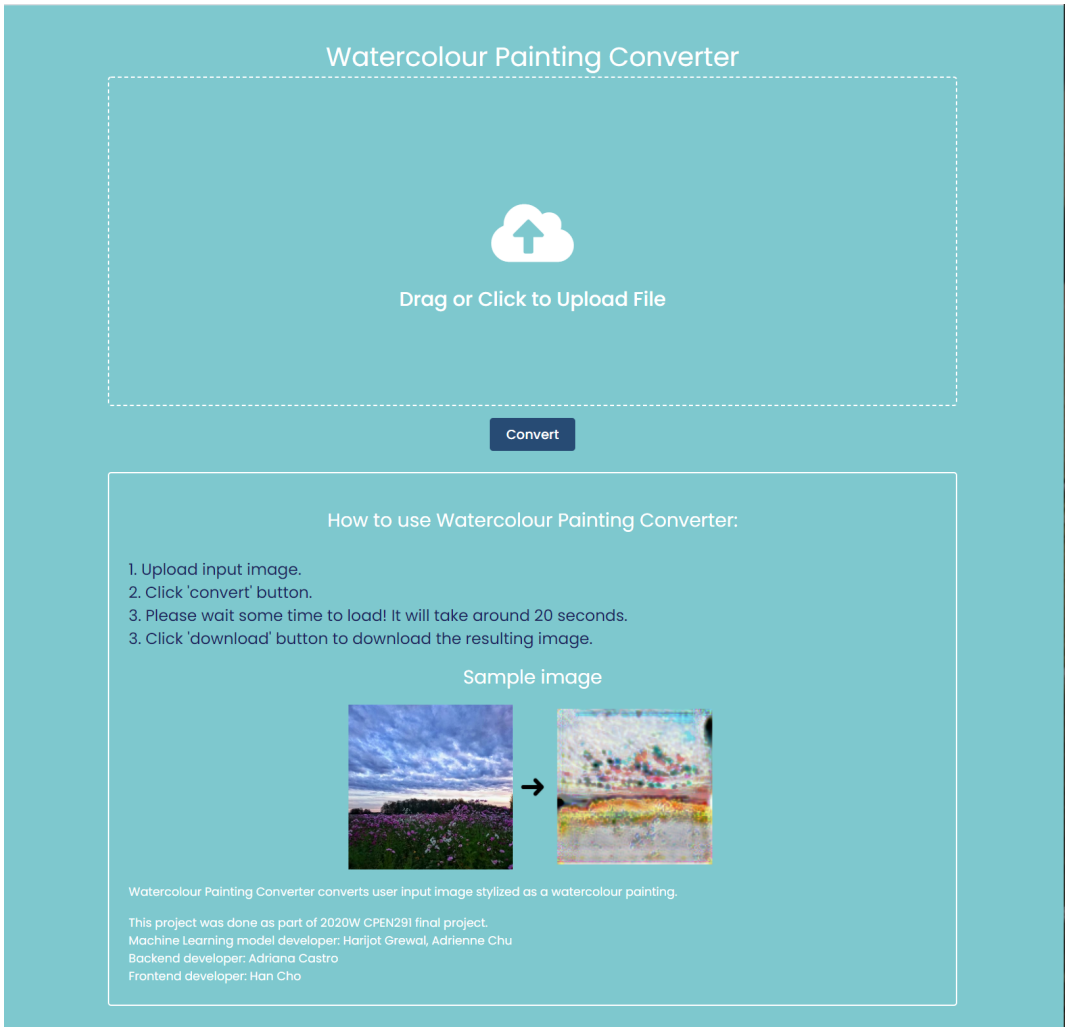
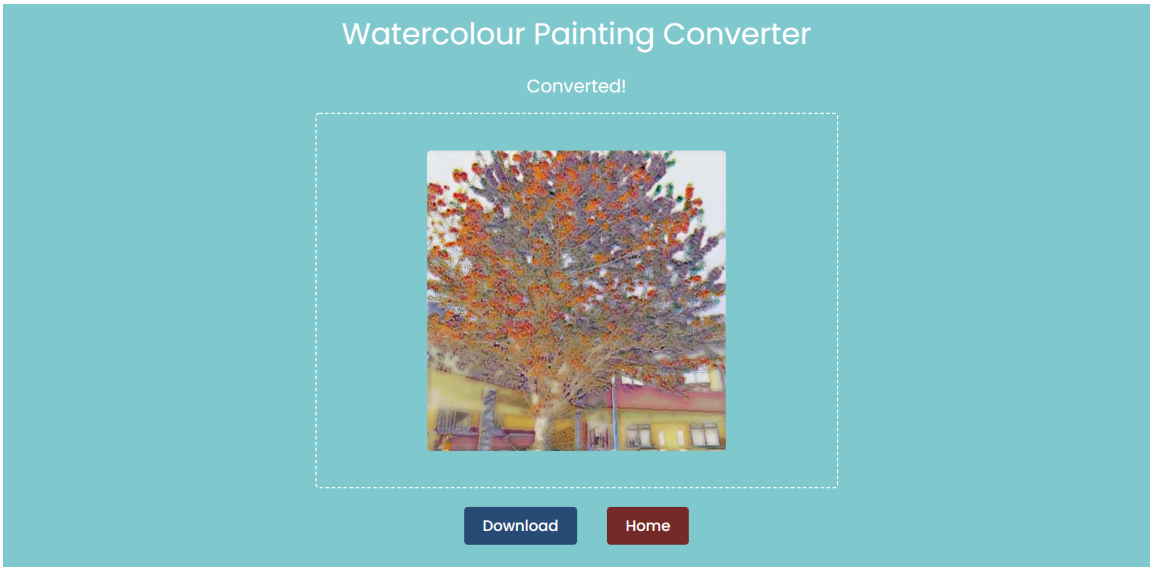
Watercolour Painting Converter Final Report

1. A brief description of the project
   "Watercolour Painting Converter" is a website where a user can upload any image and stylize the uploaded file to a watercolor painting. The frontend of the website was structured using HTML and CSS, and the ML model was created using a CycleGAN machine learning algorithm with cyclic consistency and by using several convolution layers and ResNet blocks. The backend portion connecting the frontend and ML features were written in Python using the Flask framework and Javascript.

   Overview of the Watercolour Painting Converter homepage:

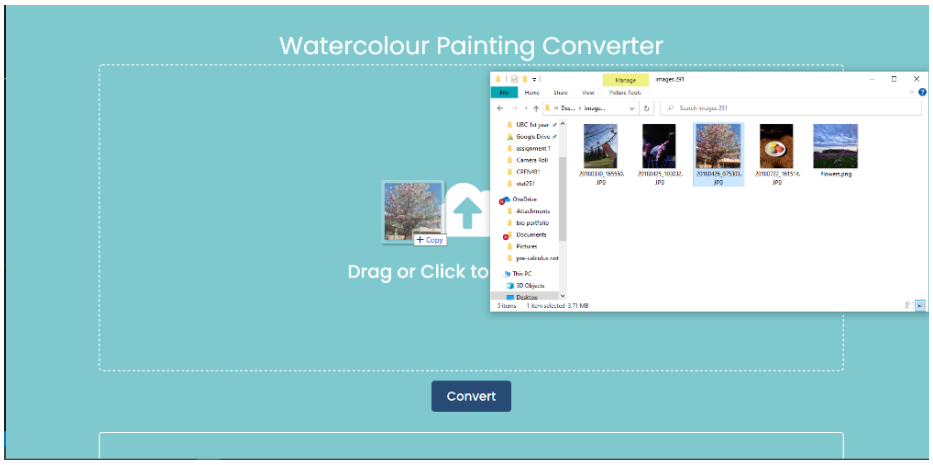


   The main website consists of title, file uploading area, convert button, and description area. Once the file is uploaded and successfully converted, it will direct to the new page that has a download button and home button as following:

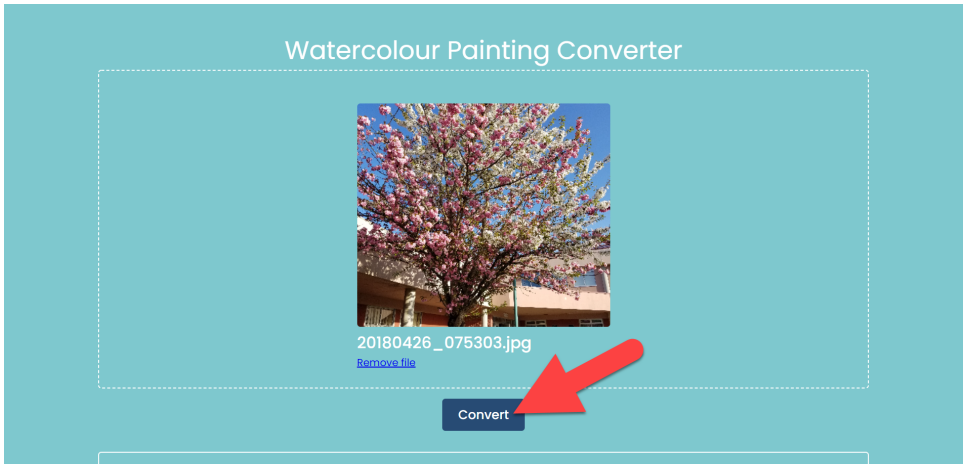


This redirected page consists of a “converted” message followed by the converted image with a download button to download the resulting image and home button to go back to the home page.

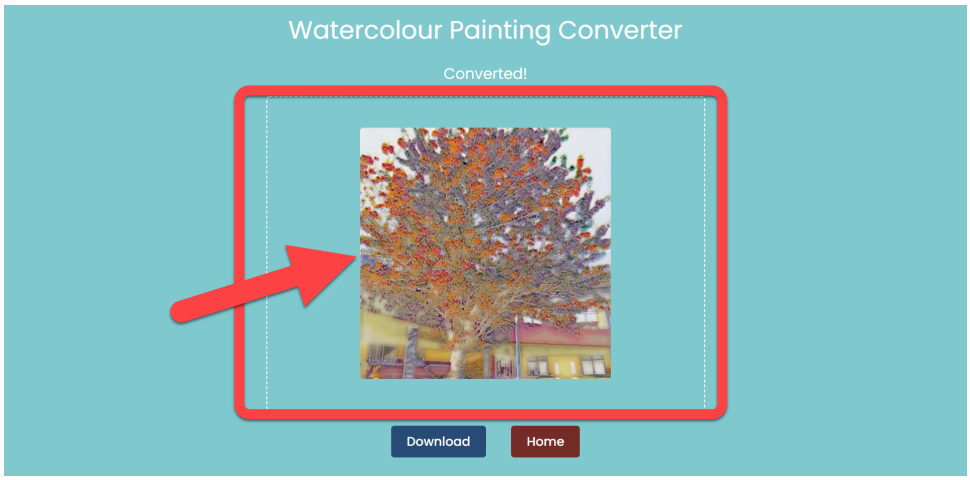
2. A walkthrough example of the project in operation
    1) Follow the README file to install the required tools and open the website (Step-by-step video is also available in github).
    2) Upload the input file by clicking then browsing or dragging the file.



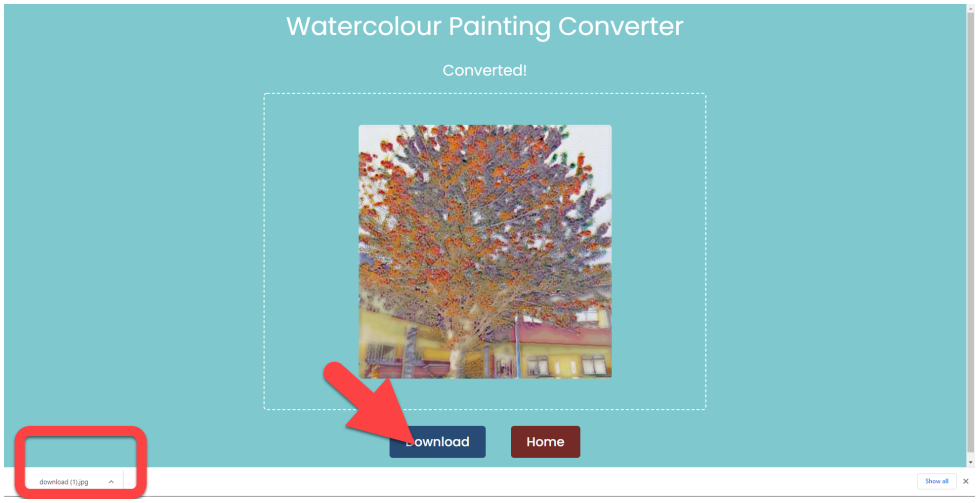
    3) Click the ‘convert’ button.



4) After some time, it will be redirected to the downloading page, consisting of ‘download’ and ‘home’ buttons.

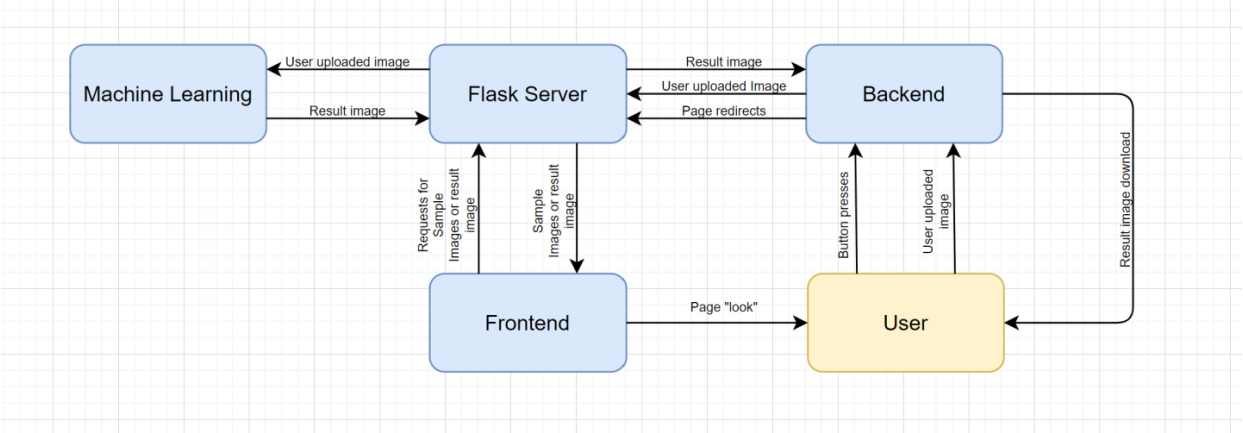


5) Click the ‘download’ button to download the converted file.



6) Click the ‘home’ button to convert another file - this will browse to the main page as shown in step 1.

3. A diagram showing all significant components in the system, plus how (and what) data flows among them.



4. A description of each component and how it was implemented (algorithms, abstractions, data structures, etc.)

Frontend component:

The website frontend component consists of designing two pages - one for the main and the other for the redirected page showing the converted result. The frontend component of this project mainly focused on the intuitive user interface. Each important section is enclosed with boxes - the border of user interactive potions such as drag-and-drop zone and download zone are represented with a dashed line, and the description area is held in a solid line box. The number of buttons is minimized, and each button is coloured differently to avoid confusion. Also, the colour and font size of the text are carefully chosen by considering their legibility.
First of all, main areas such as the area for title, drag and drop zone, description area are reserved by CSS under the HTML style tag. Each area has relative width so that they are resizing-window-friendly; formats such as width and alignment are also placed under the style tag. Then, the layout for smaller components such as text and buttons are added under each area. Under HTML body tag, those areas and features designed in CSS are gathered and structured as lego pieces.

Backend component:

Website backend is what provides the functionality of a website. For example, imagine a website with a button on it. The front end is what provides the appearance of the button – it's colour, size, shape – but the backend is what controls what happens when that button is clicked. The backend was implemented using a mixture of Javascript and dropzone.js, an open source library for creating a drag and drop area.

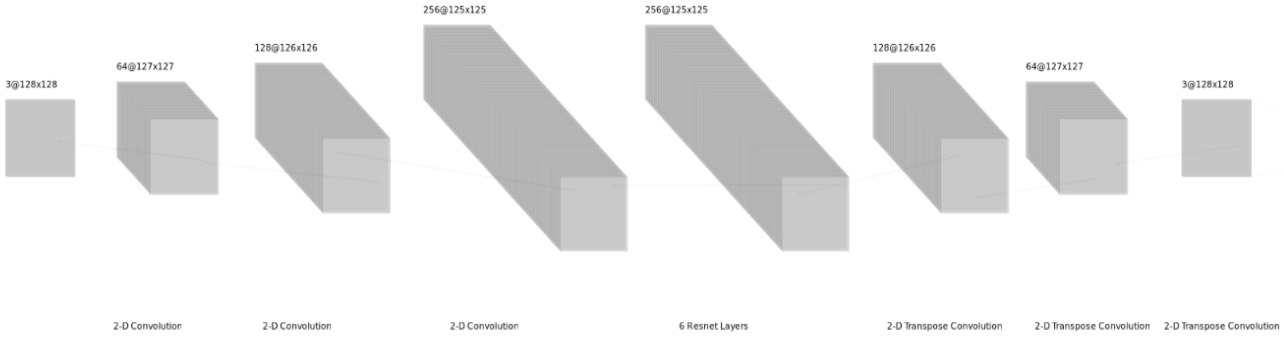
Flask Server:

The Flask server is used to specify which frontend files will be rendered for specific webpages. For example, the home page of the website for this project used WaterColourConverter.html while the '/results/' web page uses index.html. Furthermore, the Flask server is also used for image management. Uploading, deleting, or the displaying of an image on the website is all controlled by Flask. The Flask server was made using Flask and methods from the os Python library.

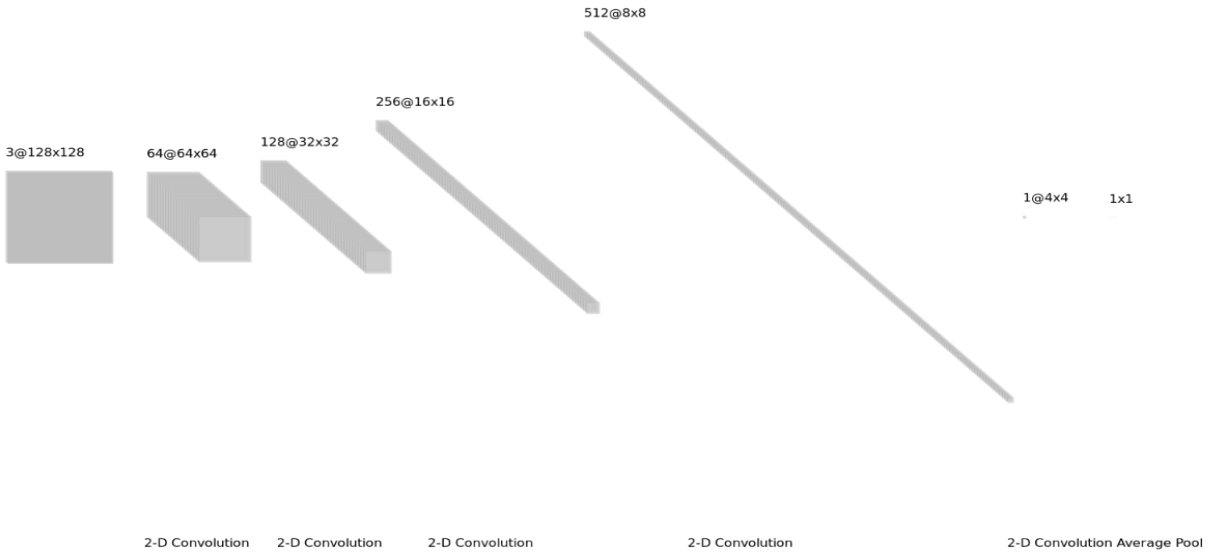
Machine Learning component:

The machine learning model used for this project was a CycleGan algorithm as a generator-critic network where there are two generator models and two critic models. The two generators are used to convert real images to watercolor images as well as convert watercolor images to real images. The critics are used to evaluate the quality of the generators by classifying the converted images as either generated or not generated. Similar to a generative adversarial network (GAN), the goal of the generators is to fool the critics into thinking the generated image is not generated. If the generator is able to fool the critic, then the generator is correctly learning how to realistically convert the image. However, by having two generators, we can implement cyclic consistency. This is done by converting the original image to a watercolor image using one generator, and then by regenerating the original image using the other. Then, we can minimize the loss between the original image and the regenerated image. Cyclic consistency allows us to perform a style transfer of one type of image onto another by forcing the generators to keep the original features of the image and only changing its style. In this case, we can transfer the style of a watercolor painting onto another image.

The overall structure for the generators is shown below going left to right with three 2-D convolutional layers, six ResNet layers, and three 2-D transpose convolutional layers.



The overall structure for the critics is shown below going left to right with five 2-D convolutional layers followed by an average pool layer.



5. The overall contributions of each team member to the project. This needs less detail than the milestone reports, we just want to know who was responsible for which parts
    a. Han Cho: Frontend of the website
    b. Adriana Castro: Backend of the website
    c. Harjot Grewal: Machine Learning Model construction and training portion.
    d. Adrienne Chu: Data scraping portion and training portion of Machine Learning Model.