**Solution design**

Solution Description:

Our solution would aim to use the recently developed, novel technology, called GANs, that would take colorized images, turn them to black & white and then train itself on those images and see how accurate it is by comparing itself to the original image and use it for training. We plan to use adaptive learning rate to speed up the training process. One of the approaches that we’re experimenting with is called “Decaying Learning Rate” where the model starts with a very large learning rate and then slowly starts decaying with each iteration. After the training is finished it should be possible to input any image that is originally black & white and convert it to a colorized picture with colors filled out looking like it was colored by a human.

Data Description:  
 For our baseline model we’ll be using pictures from publicly available datasets containing pictures to see if what we are trying to do is the correct approach.

After that, we will use multiple copyright free images from an online repositories of royalty-free images, as well as datasets handpicked for machine learning modules to train our model.

“<https://www.kaggle.com/shravankumar9892/image-colorization/data>” is an image dataset that was prepared for Image Colorization and consists of 25 thousand pictures, 224 pixels wide and 224 pixels high. They are spread into two categories, the original pictures and the black & white images which makes it easy to see accuracy and how the model performs. Their small and consistent size makes it easy to train and prepare the neural network for. What is more, those images are stored in .npy files, which are known as one of the fastest files to contain data in, outperforming .txt and .json files greatly.

“<https://storage.googleapis.com/openimages/web/index.html>” is Google’s repository of thousands upon thousands of images of all kinds. They are mostly colorized so we’ll turn them black & white ourselves but they are of varying sizes and content that could prove useful when making our project accessible to people who can have photos with different sizes and content.

“Unsplash” — creative commons pictures by professional photographers. It includes 9,500 training images and 500 validation images. Potentially good dataset, could be quiet small for our needs.

Having said all of that, and despite all of these datasets being great, all of them have some sort of downsides. For our baseline, I decided to go with a dataset made by Emil Wallner. His dataset contains about 9.5 thousand pictures, with 500 more for testing. They are generally focused on humans (as in, they portray human beings), which I feel should be our aim when training the model, but also have additional images that can be seen as more abstract, ensuring that whatever image is given, it should be colorized correctly. This dataset can be found at: <https://www.floydhub.com/emilwallner/datasets/colornet>

Our model avoids any and all ethical and legal issues that use of random images could potentially cause since people who compiled datasets already went through all of those issues and compiled a proper dataset.

Solution Motivation:

Reasoning for choosing this solution:

* More efficient than coloring pictures by hand.
* Allows for theoretically higher accuracy than amateur human colorizers.
* Faster conversion – (close to) instantaneous, in contrast of years it could potentially take for a human to completely research the time period to accurately colorize the image, which even then, could prove inaccurate.
* Potential for future implementation for videos. Movies are (usually) 24 pictures (frames) per second, so with enough training data, and GANs, we could potentially colorize movies from a time period of black & white movies. This, of course, is not the main priority and can end up being part of our continuous development plan.
* Can see niche use in other fields. For example, NASA’s “picture of the day” section is usually filled with images that had to be edited by a human, our program could end up saving time and money. Example 2: Medical images (xray, scans) are usually very barebones since they are supposed to convey information rather than look pleasingly which can make them look intimidating. Our module could introduce colour to those images to hopefully increase patient’s engagement into the medical process or even entice people to join the medical field.
* Possible for a breakthrough in the AI field. There is some research done around this topic, but our approach could end up uncovering some breakthrough approach, solution or even model accuracy that can end up being industry standard.
* There is a need and demand for something like that (reddit’s r/estoration subreddit related to photo restoration & colorization that has 84k active members).
* We could impact people’s lives by literally bringing colour into their memories.

ALPHA ALGORITHM STEPS – OUR BASELINE – NO COMPLEX ALGORITHM:

1. Acquire the dataset – a single picture and save it in the project’s folder.
2. Import the required libraries (In baseline of a baseline it’s: keras, tensorflow for backend, numpy for working with matrices, skimage for working with images – converting colorscales, os for working with directories and files)
3. First of all, load the 400px x 400px image and save it into an array/matrix/tensor, where each pixel is represented by values in the matrix.
4. After that, convert the RGB color scale into lab values.
5. Normalize the data (flatten it) and reshape it.
6. Create layers of the neural network (do some upsampling and convolution 2d layers)
7. Compile the model.
8. Begin the training session – 1000 epoch is pretty much minimum for a decent cold color scale picture whereas 100 is sometimes enough for warm color scale. Batch size of 1 because there’s only 1 picture.
9. Print out the evaluation of the model and put the output back into normal, unflatten data (by multiplying it by 128 – where we had to divide it previously to work well with the neural network).
10. Finally, save the created picture ( convert the lab color scale into rgb) as well as the grey version for comparison (lab -> rgb, rgb-> gray).

ALGORITHM STEPS – THE INITIAL PLAN OF THE FINAL PRODUCT

1. Acquire the data set, see if pictures exist in the folder (Train/) by default, notify the user if they don’t.
2. Load the entire folder of pictures into their respective array.
3. Convert their RGB color scale into Lab values and extract Lightness channel.
4. Neural network – Use kernels to flatten the data, conduct 2d convolutions.
5. Compile the model.
6. Begin the training session – Batch size and steps\_per\_epoch should generally be equal (800 is ok, load more if there is enough memory), decaying learning rate.
7. Keep on training till you go through the entire folder of data or till the performance starts to drop down (not implemented - was supposed to be part of team mission).
8. Save the model.
9. Test the neural network using about 10% of the loaded pictures, show how accurate it(loss value) was and some technical details.
10. Go through the “Test/” folder and colorize some pictures using the model's prediction. (this is done since it’s better to see what the model can actually produce, rather than seeing some numbers, which can be inaccurate)
11. Save the colorized pictures.

OUR MODEL FROM CONSUMER’S POINT OF VIEW - GUI:

1. User has a black & white picture they want to colorize.
2. Open up our program.
3. Select option to put the file into the program
4. Wait for the program to convert it and save it for you.
5. Check out the produced photo.

OUR MODEL FROM CONSUMER’S POINT OF VIEW - command line:

1. User puts the picture they want to colorize in the working directory of the program (default folder: “Test/”)
2. They run “python colorize.py” once the dataset is placed correctly.
3. The model and the picture(s) are loaded in.
4. Model’s prediction is run across the picture(s) given by the user.
5. Unflatten and unnormalize the pictures so they can be saved without losing any information.
6. Save the colorized pictures.

