Deliverable 2

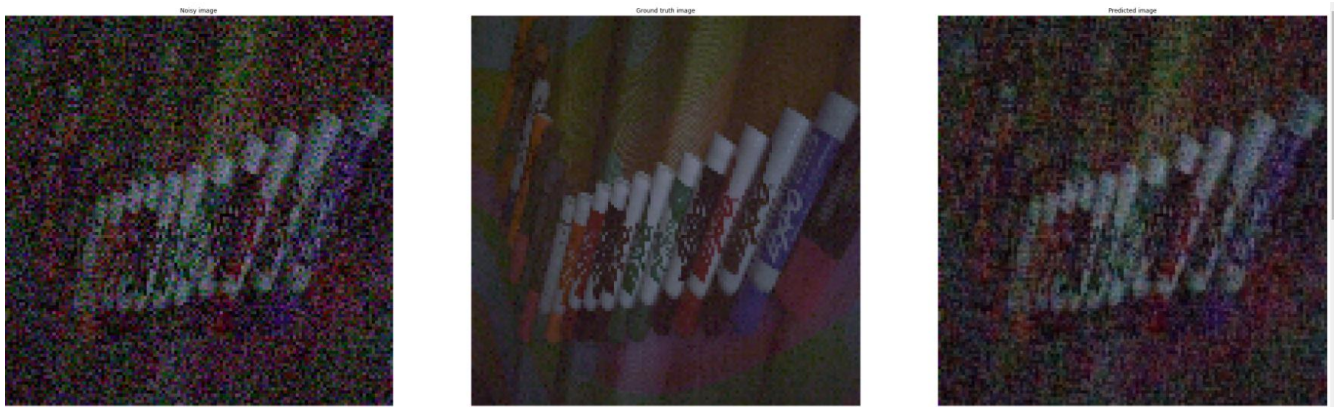
1. This project is about denoising RGB images. Noise is a random variation of brightness or color information in images, which we are trying to reduce.
2. I am currently using the SSID small dataset:
   https://www.eecs.yorku.ca/~kamel/sidd/dataset.php
   For now, the only preprocessing that I did was getting the path of all the images into arrays, and then loading the images from the list of paths. In my deliverable 1 I said that I would split the images into smaller $n \times n$ images, but for now I didn't do it because I am bit late one this deliverable, and I didn't have enough space to store the split up images on my computer, and I saw that denoising was also done one the full images. I will still do it to see how this will change the effectiveness of my model, and change the value of n to find the best possible value. I also originally thought of splitting the images into 3 sub images, red, green and blue, but this is taken care of in the model, so no need to for preprocessing here.

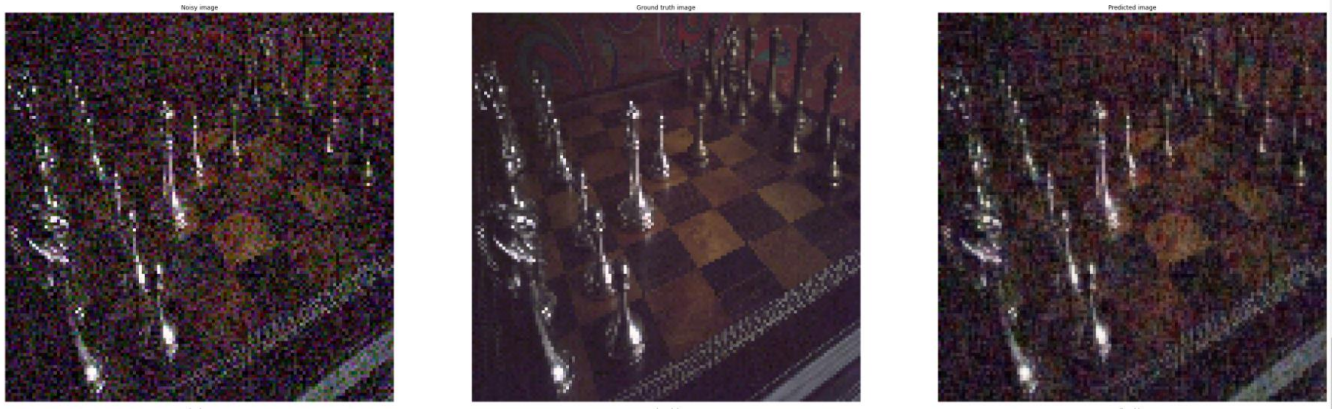
3. In my deliverable 1, I wasn't completely sure on what model I should use, but after reading about denoising (and Thomas suggesting it too), the model that I chose was convolution neural networks autoencoders (CNN). This seemed like the simplest model to implement, especially since there were many resources explaining the model and showing implementations.

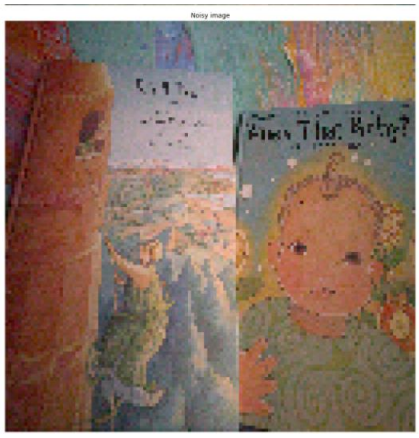
   For my model I mainly used tf.keras for the CNN implementation and cv2 for the image processing part. I chose ts.keras for CNN because this is what most resources on the internet used. I am not familiar at all with CNN, so this seemed like the best library to use. For splitting my test, and train sets, I used sklearn.model_selection.train_test_split(), and split my 160 images dataset into 90% for test and 10% for train. The only "optimization tricks" that I used would be using np.arrays instead of python lists. For hyperparameter I haven't really tried a lot. The hyperparameters that I'll be playing with are what tf.keras provides in the Conv2D function, and the size of the smaller images n when I try splitting the images. I can also try adding more convolution layers in the CNN, but I'm not sure if this could be considered a hyperparameter.

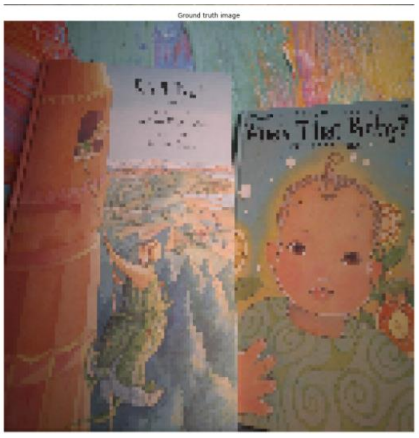
   For testing my model, I used the PSNR (peak signal-to-noise ratio) metric, as well as the SSIM (structural similarity index measure) on my X_test and y_test sets
   The average PSNR for the noisy images was 29.436035.
   The average PSNR for the predicted images was 31.212303.
   The average SSIM for the noisy images was 0.756652.
   The average SSIM for the predicted images was 0.841863.
   The PSNR value didn't have a really big increase, but from what I read this wasn't too bad. The SSIM had a pretty good increase though.

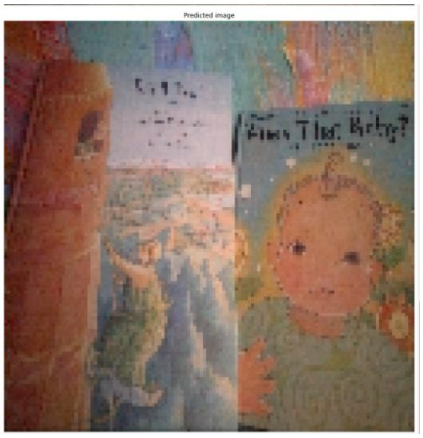
But both metrics aren't always good objective measures for predicting human visual response to image quality. And from research, those 2 are considered the best metrics available for image processing.
I would say my model may be close to become underfitting. It is working (seeing from the measurements, and from the predicted images themselves) but the difference isn't that big compared to the noisy images. I have included some of my results. The left is the noisy, middle is the ground truth and right is the predicted image (I will explain later why all those pictures have bad resolution).











I would say the results aren't that bad but aren't great either. I also chose the pictures that had a visible difference between the noisy and the predicted. Many other of my results didn't differ that much from the noisy image.

I faced 2 main challenges in this project.

First was trying to understand the CNN, which I still don't fully understand. After reading a lot about them, and seeing different implementations of them, and their applications in image denoising and image super resolution I got an idea about how they work, but not a good understanding.

Second was the limitation of google collabs. I don't know if this is supposed to happen, or my code isn't good, but when I first tried implementing the CNN, I would get an error that I used all the ram available to me (which is around 12-13 gb). So I had to resize my images to 128*128, which is why the quality is really bad on the images above. I also had to reduce the number of convolutional layers (I don't know how much this affected ram though, or if it even did do anything, I need more testing to make sure). So I'm assuming if I could work with higher quality images, I'd get a better working neural network. I think I'll have to try doing it on my pc, which

also doesn't have that much ram (16 gb), but here I assume this will only slow down the execution, not stop it altogether like in google collab.

As for graphs, I don't have any because I don't know what I should put in a graph for this specific project. I already talked about the 2 evaluations metrics that I used, but I doubt using them in a graph would be useful. The only thing I could think of would be using my trained model on multiple datasets, focusing on some varying variable in the datasets, and then graph my metrics results. For example, I could work on datasets of varying image brightness, rank the dataset by increasing average brightness of the images, then us my model, and plot the resulting metrics. This would help see how my algorithm perform on different degrees of brightness (so the degree of brightness would be the varying variable). I don't know if I could find such datasets, but if I can this would help generalize the algorithm to all images, and if a dataset isn't performing well I could use it in the training.

In conclusion, I think my model is performing well for a start. To improve it I'll:

1. Add more images to the training sets
2. Split the images into smaller images
3. Try to maximize the allowed resolution to not run out of RAM
4. Try changing some hyperparameters