Clairvoyance: An automated image post-processing tool

With subject identification, image classification, and generative correction

An Introduction to Machine Intelligence Project Final Report

Github link to project: https://github.com/mie324/Clairvoyance

Authors:
Kevin (Gen Sheng) Zhang
Morris Jamieson Chen

December 2nd 2018
For Prof. Jonathan Rose

Word Count: 1948
(Penalty: NULL)

Introduction

Photographers, both professionals and casuals, invest huge amounts of time doing *identification*, *classification*, and *correction*. The post-capture workflow of photo-takers can be automated using several different neural networks trained to perform the different tasks. From first-hand experience, popular photo-editing tools such as Adobe Lightroom and Photoshop fail to meet the demand for smarter "one-click" image filtering and enhancement. Editors wish to select and retouch photos in their preferred style, but often face a trade-off between time and quality when auto-tuning from existing software yields lacklustre results.

The goals of Clairvoyance are to *identify*, *classify*, and *correct* photos. To limit the scope of this project, human subjects (faces) will be the focus to make *identification* easier. The photos will be *classified* based on three metrics [1], which are common problems photographers face:

1) Focus (unfocused photos are nearly impossible to correct)
2) Noise & Artifacts (difficult to correct without losing brightness or quality)
3) Lighting (usually correctable in less compressed formats)

And, finally, *correction* will be applied in the same three categories introduced above. In just one month, Clairvoyance was built from scratch to be a "one-click" workflow assistant that fulfills these goals to a great extent.

Overall Software Structure

The Clairvoyance software is a desktop application that is command-line executable. The program was written using an object-oriented approach and multiple instance methods to improve organization and readability. Figure 1 shows the overall software structure, and Figure 2 shows the final input and output directories.

Initialization

The program takes in several inputs and saves them as attributes:

"model": The image quality assessor (IQA) model used to generate quality predictions.

"tolerances": A list of three numerical values between 0 and 1 [*focus, noise, lighting*] that dictate how tough the selection of images is based on predicted results.

"input_dir": The path to input directory containing images to-be-*clairvoyanced*

"output_dir": Optional parameter for path to output-directory

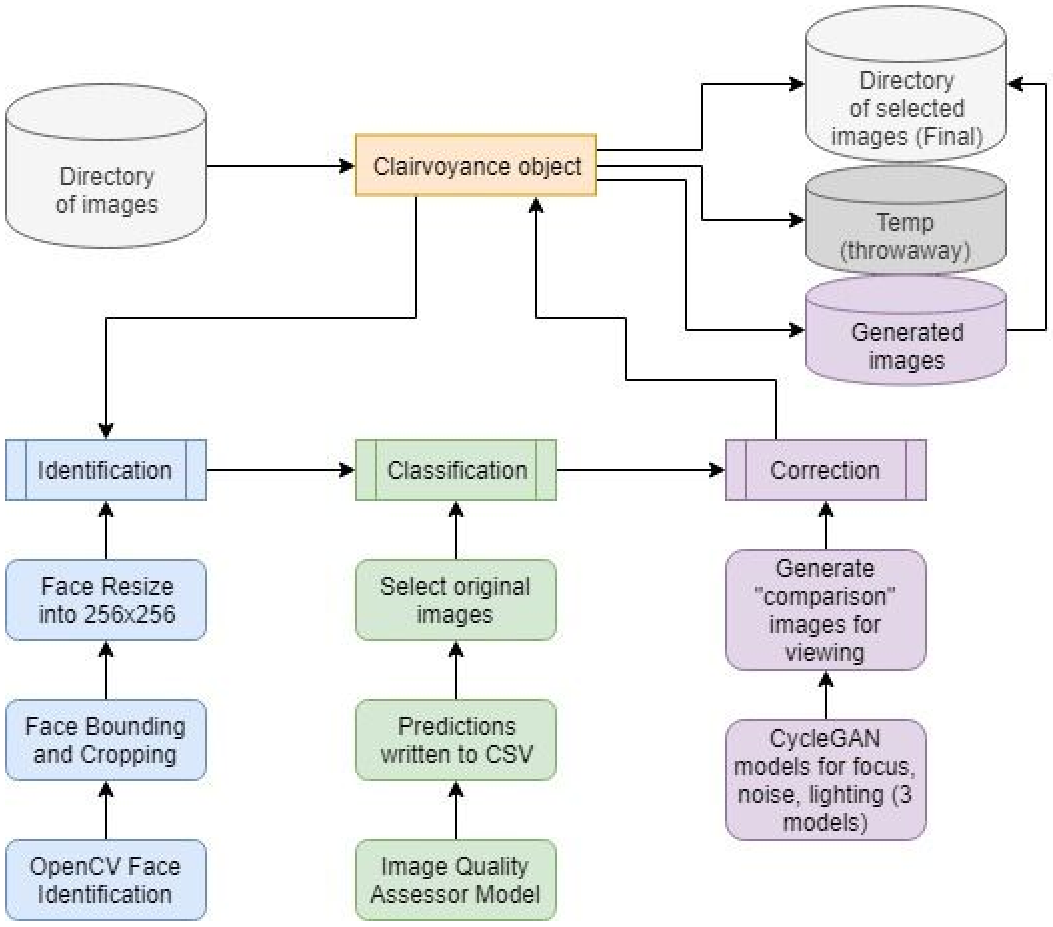

Directory of images
Clairvoyance object
Directory of selected images (Final)
Temp (throwaway)
Generated images
Identification
Classification
Correction
Face Resize into 256x256
Face Bounding and Cropping
OpenCV Face Identification
Select original images
Predictions written to CSV
Image Quality Assessor Model
Generate "comparison" images for viewing
CycleGAN models for focus, noise, lighting (3 models)

Figure 1: Software Flow for Clairvoyance

<u>Identification</u>

Clairvoyance uses OpenCV's cascade object detector, with the pre-trained `haarcascade_frontalface_alt` classifier for identifying faces [2]. The idea is to use a sliding-window feature-based approach with the classifier as a search window, and look for specific edge, line, and center features. "Cascade" means that the classifier is a composition of several simpler classifiers, each of which can be a search window. Although similar to how a convolutional neural network (CNN) may operate, this method does not employ a NN. After the face is identified, it is cropped, resized to 256x256px, and saved in *temp*.

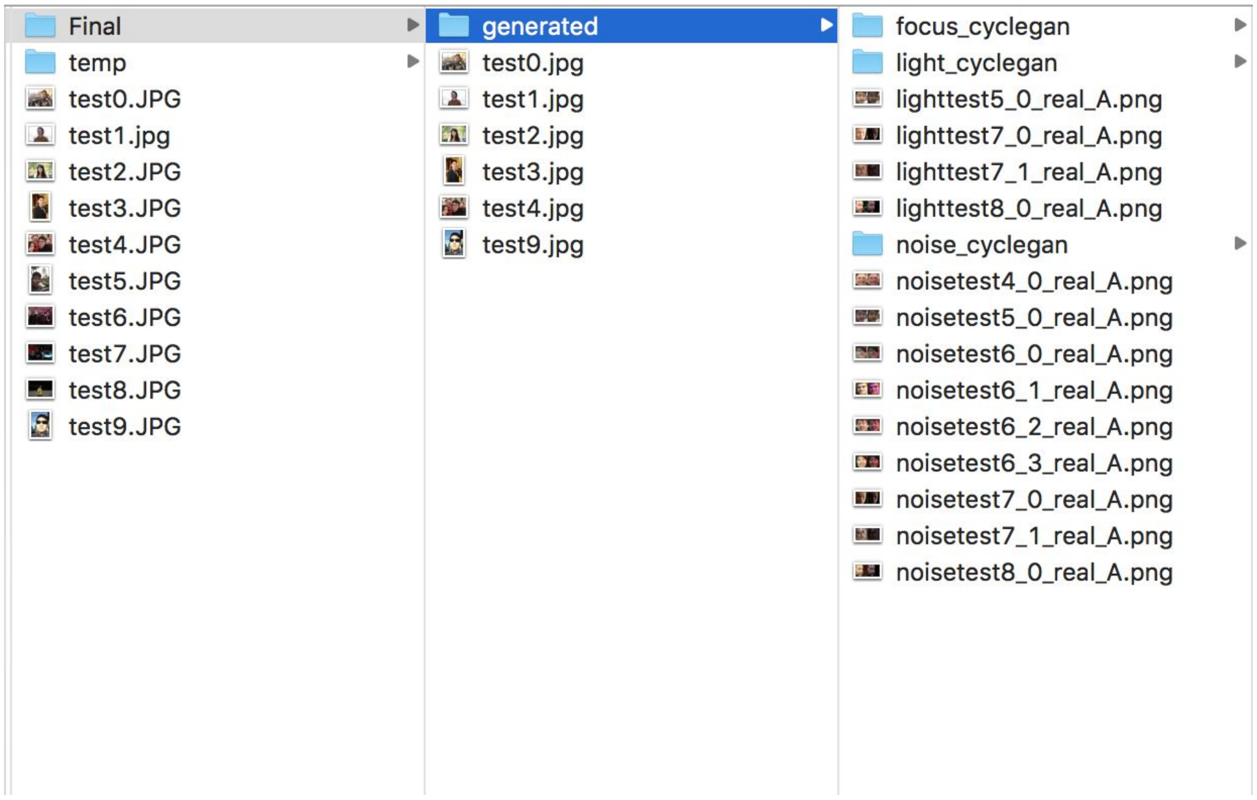
The focus of Clairvoyance is not face detection, as it will eventually be expanded to work with any subject. Using OpenCV, time was saved to explore the more interesting NNs.

<u>Classification</u>

The IQA is a CNN that performs regression on an image to determine the three quality parameters of focus, noise, and lighting. The Clairvoyance IQA concept is rather unique in the sense that it returns a multivariate output that provides an evaluation of the image in three different respects, rather than a single score. Users are able to provide different tolerance values depending on their preference. Although the IQA selects based on subject quality alone to tackle issues like Bokeh and night-scenes [3], the classification step selects the entire original image and saves it in *Final*.

<u>Correction</u>

In this final step, CycleGAN models are used to generate improved versions of the images that did not pass classification. Again, based on which parameters did not make the tolerance, improvement is attempted using one or more of the three models. This way, it is possible to improve images in only one aspect, or multiple aspects in series. Due to the work on cropping and resizing, it is difficult to stitch the generated image into the original image while maintaining a harmonious look. In the future, the GAN could be used to improve the entire original image, but that was not possible due to limited training time and computation power. As an intermediate step, the generated results are saved alongside the originals in *Final*⟶ *generated* for users.



Final
temp
test0.JPG
test1.jpg
test2.JPG
test3.JPG
test4.JPG
test5.JPG
test6.JPG
test7.JPG
test8.JPG
test9.JPG

generated
test0.jpg
test1.jpg
test2.jpg
test3.jpg
test4.jpg
test9.jpg

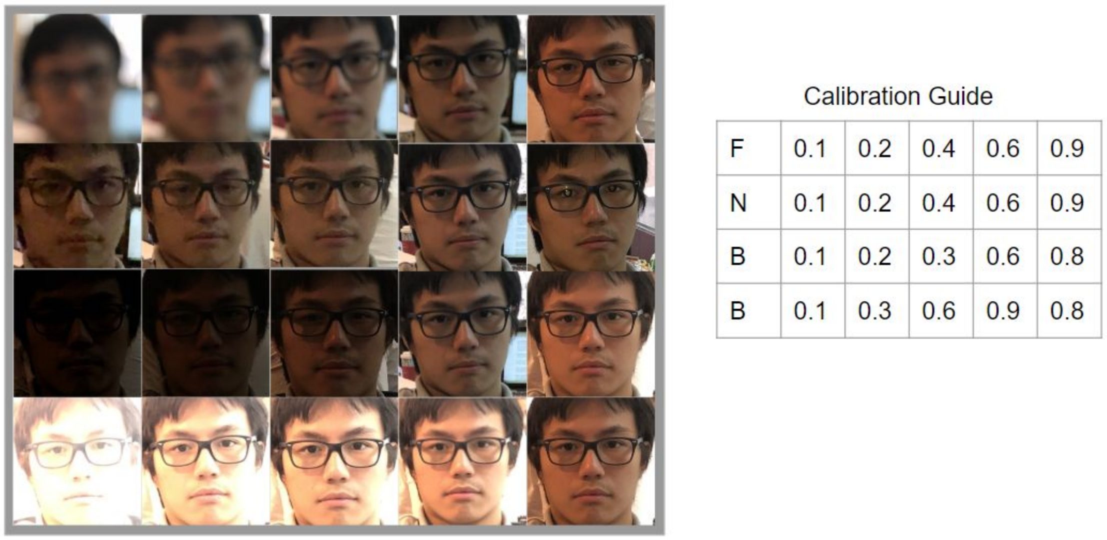
focus_cyclegan
light_cyclegan
lighttest5_0_real_A.png
lighttest7_0_real_A.png
lighttest7_1_real_A.png
lighttest8_0_real_A.png
noise_cyclegan
noisetest4_0_real_A.png
noisetest5_0_real_A.png
noisetest6_0_real_A.png
noisetest6_1_real_A.png
noisetest6_2_real_A.png
noisetest6_3_real_A.png
noisetest7_0_real_A.png
noisetest7_1_real_A.png
noisetest8_0_real_A.png

Figure 2: Final Directory Layout for Clairvoyance on test 1-10, 0-4 and 9 are selected, and rest are improved based on their quality parameters in generated. Left⟶Right: Input Dir, Final, generated

Sources of Data

A repository of about 3000 photos was compiled from entirely personal resources. These include photos taken with multiple DSLRs, point-and-shoots, and phone cameras. For each photo, faces were detected, cropped, resized, and saved. Then, numerical labels were assigned manually for each face for: focus, noise, and lighting, 0 for being unacceptable and 1 for being exemplary.

To prevent the labels from becoming too subjective, a calibration chart (Figure 3) was used.

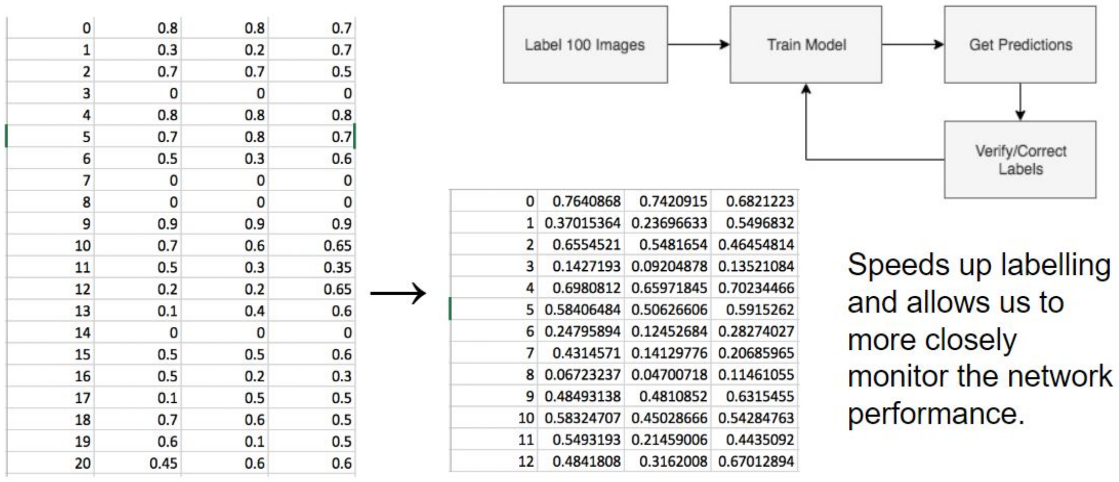


Calibration Guide

F	0.1	0.2	0.4	0.6	0.9
N	0.1	0.2	0.4	0.6	0.9
B	0.1	0.2	0.3	0.6	0.8
B	0.1	0.3	0.6	0.9	0.8

Figure 3: Calibration Chart for Labelling Images

An assisted labeling technique was also employed to speed up the labeling process. It involves first labelling a small sample of photos, training an IQA model based on this and getting predicted results, and then adjusting the predicted results to be used for training again. Shown in Figure 4, this method saves time and effort necessary to label each new image by introducing a constantly improving point of reference.



0	0.8	0.8	0.7
1	0.3	0.2	0.7
2	0.7	0.7	0.5
3	0	0	0
4	0.8	0.8	0.8
5	0.7	0.8	0.7
6	0.5	0.3	0.6
7	0	0	0
8	0	0	0
9	0.9	0.9	0.9
10	0.7	0.6	0.65
11	0.5	0.3	0.35
12	0.2	0.2	0.65
13	0.1	0.4	0.6
14	0	0	0
15	0.5	0.5	0.6
16	0.5	0.2	0.3
17	0.1	0.5	0.5
18	0.7	0.6	0.5
19	0.6	0.1	0.5
20	0.45	0.6	0.6

$\longrightarrow$

0	0.7640868	0.7420915	0.6821223
1	0.37015364	0.23696633	0.5496832
2	0.6554521	0.5481654	0.46454814
3	0.1427193	0.09204878	0.13521084
4	0.6980812	0.65971845	0.70234466
5	0.58406484	0.50626606	0.5915262
6	0.24795894	0.12452684	0.28274027
7	0.4314571	0.14129776	0.20685965
8	0.06723237	0.04700718	0.11461055
9	0.48493138	0.4810852	0.6315455
10	0.58324707	0.45028666	0.54284763
11	0.5493193	0.21459006	0.4435092
12	0.4841808	0.3162008	0.67012894

Speeds up labelling and allows us to more closely monitor the network performance.

Figure 4: Slide from Progress Presentation on Automated Data Labelling

The end result is that the IQA-training dataset was formed from scratch, with about 1000 labelled images, and the result from the best IQA model was used to split images, which formed the datasets for the CycleGAN.

Machine Learning Models

<u>Image Quality Assessor</u>

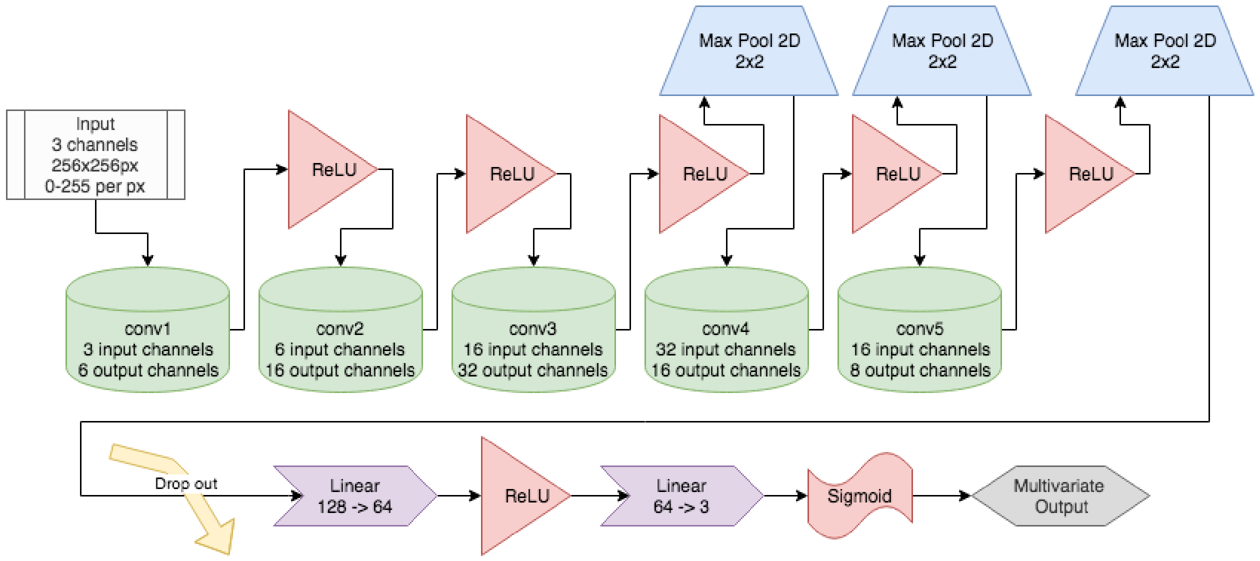


Figure 5: Model architecture for the CNN Regressor used for IQA (U-Net Inspired)

Figure 5 shows the best attempted CNN design for IQA in Clairvoyance. It takes in a 3 channel, 256x256 input and undergoes 5 convolution layers, 3 max pools, a drop-out, and 2 linear layers to result in a size-3 multivariate output for the quality parameters. The model was constructed based on the U-Net research paper, with a contracting path with increasing kernel sizes and number of channels to capture context, and a symmetric expanding path with decreasing kernel sizes and number of channels to enable precise localization [4]. Max-pooling was used on the expanding path to discretize the data and take into account only the most significant features, as well as reduce the model size and hence training time. LeakyReLU was used to allow a small non-zero gradient when the unit is not active to minimize the number of *dead* neurons [5]:

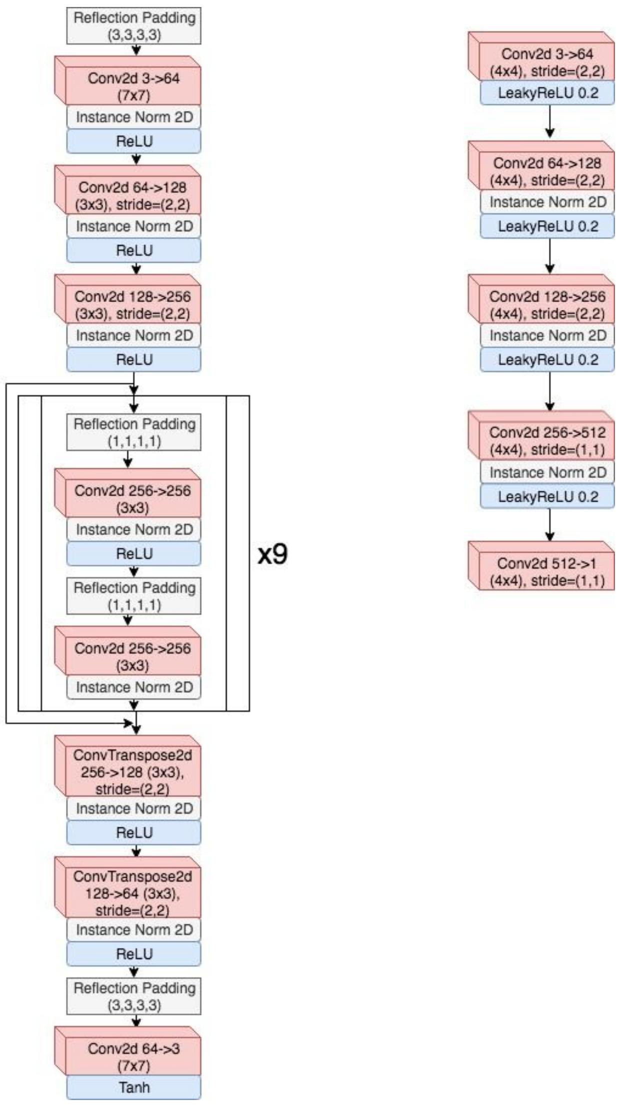
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$

Also, dropout is used as as *regularization* technique to randomly ignore the state of certain neurons during each epoch of training. This helps to mitigate over-fitting, by reducing inter-dependent learning amongst neurons and encourages the learning of more robust features and less random variations [6].

Finally, the linear layers maps the convolution layers' output to a size-3 multivariate output, through a standard voting procedure. The final pass is through a sigmoid activation function to restrict outputs between 0 and 1.

<u>CycleGAN</u>

The CycleGAN architecture was used to perform super-resolution on images. Although the standard problem proposed by the original paper [7] is to perform domain transfer between two unpaired classes of images (i.e. horses to zebras), the Clairvoyance project investigated the feasibility of training an unsupervised super-resolution model. The CycleGAN model involves two adversarial networks, each converting images of either class to the other. The Resnet 9-blocks and PatchGAN [8] networks were used for the generators and discriminators, respectively.



Reflection Padding (3,3,3,3)
Conv2d 3->64 (7x7)
Instance Norm 2D
ReLU
Conv2d 64->128 (3x3), stride=(2,2)
Instance Norm 2D
ReLU
Conv2d 128->256 (3x3), stride=(2,2)
Instance Norm 2D
ReLU
Reflection Padding (1,1,1,1)
Conv2d 256->256 (3x3)
Instance Norm 2D
ReLU
Reflection Padding (1,1,1,1)
Conv2d 256->256 (3x3)
Instance Norm 2D
x9
ConvTranspose2d 256->128 (3x3), stride=(2,2)
Instance Norm 2D
ReLU
ConvTranspose2d 128->64 (3x3), stride=(2,2)
Instance Norm 2D
ReLU
Reflection Padding (3,3,3,3)
Conv2d 64->3 (7x7)
Tanh
Conv2d 3->64 (4x4), stride=(2,2)
LeakyReLU 0.2
Conv2d 64->128 (4x4), stride=(2,2)
Instance Norm 2D
LeakyReLU 0.2
Conv2d 128->256 (4x4), stride=(2,2)
Instance Norm 2D
LeakyReLU 0.2
Conv2d 256->512 (4x4), stride=(1,1)
Instance Norm 2D
LeakyReLU 0.2
Conv2d 512->1 (4x4), stride=(1,1)

Figure 6: The Resnet 9-blocks generator (left) and PatchGAN discriminator (right) were used for each direction of domain transfer.

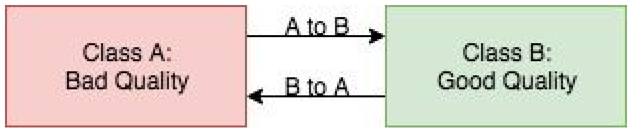


Figure 7: The overall flow of CycleGAN involves two generators, one in the forwards direction, and the other in the reverse.

The rationale behind creating the reverse generator is to enforce cycle-consistency. Because the nature of the forwards generator is to create an image of a higher quality based on a low quality image, the generator may completely alter the contents of the image. Having a loss function between a recreated image and the original image mitigates this problem by restricting the range of outputs that may be undesirable for super-resolution.

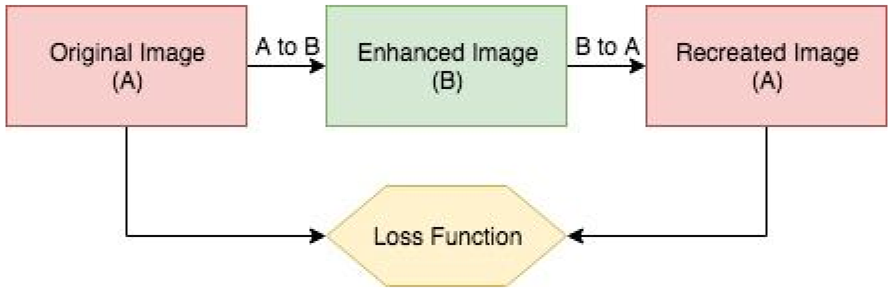
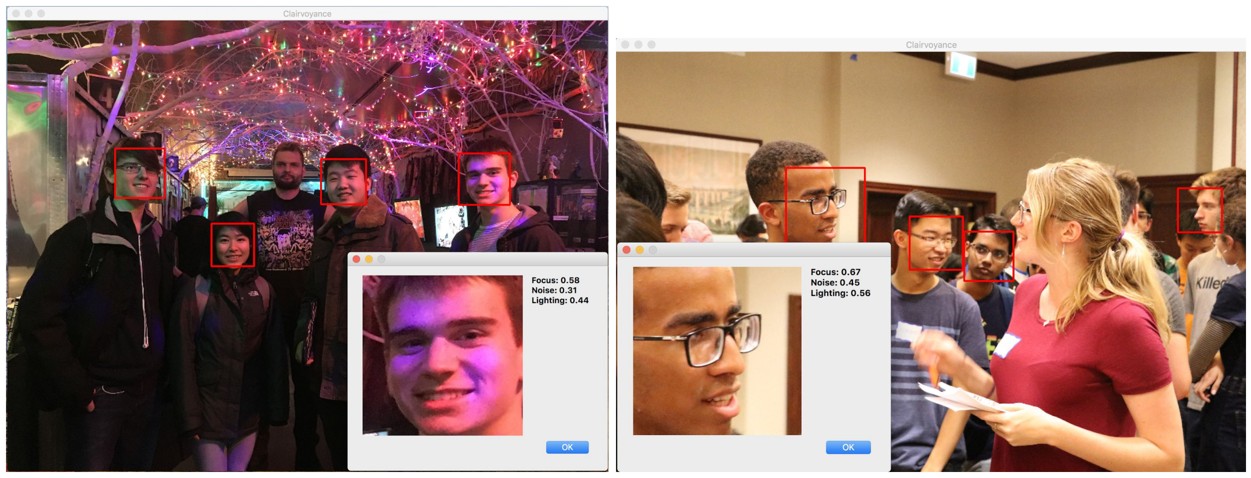


Figure 8: The recreated image is generated using the reverse generator so that a loss can be computed with the original image.

The cycle-consistent property of the CycleGAN is utilized to ensure that the contents of the image after the super-resolution remain the same. The CycleGAN model Applying certain thresholds to the predictions from the IQA model, two classes of low and high quality images were created for each of the three quality parameters: focus, noise, and lighting, resulting in 3 generative models.

Training, Validation, Test Results

Image Quality Assessor



Focus: 0.58
Noise: 0.31
Lighting: 0.44

Focus: 0.67
Noise: 0.45
Lighting: 0.56

Figure 9: IQA Brower

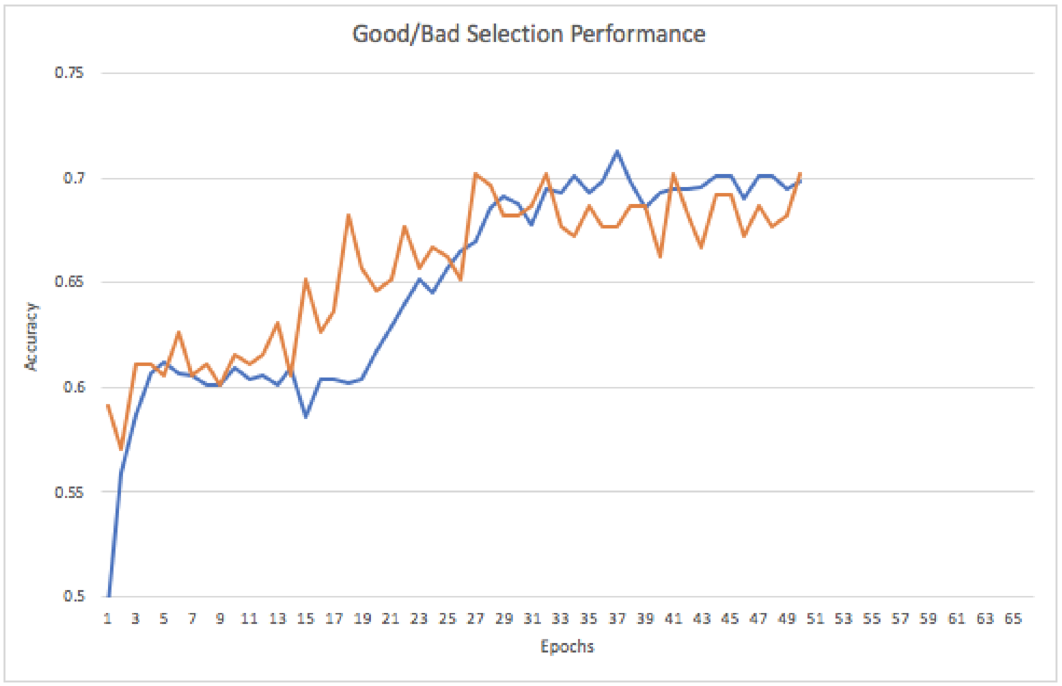


Figure 10: IQA performance for determining good/bad photos with [0.5 0.5 0.5] tolerance
Legend: Blue - Training, Orange - Validation

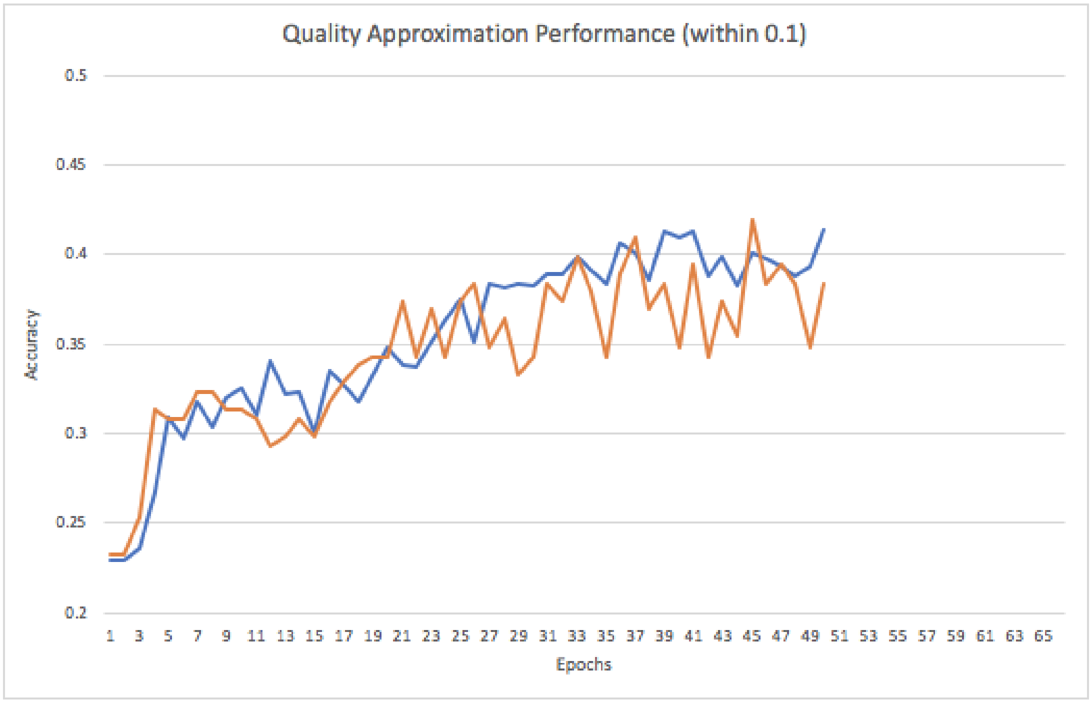
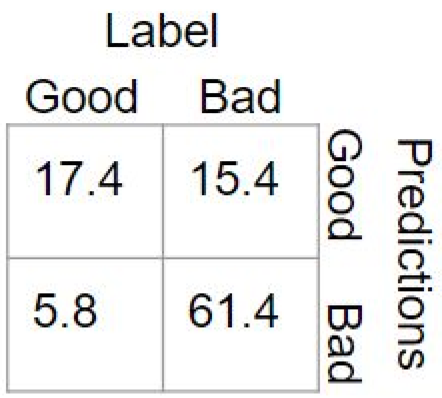


Figure 11: IQA performance for closeness in magnitude compared to label (<10%)
Legend: Blue - Training, Orange - Validation



	Label: Good	Label: Bad
Predictions: Good	17.4	15.4
Predictions: Bad	5.8	61.4

Figure 12: Confusion matrix for IQA on test dataset of 175 images

The Adam optimizer was used for training because it yields considerably faster results without too much observable convergence difference when compared with SGD. The specific technical reasons for this can be seen in the Adam Stochastic Optimization paper [8].

Figure 9 shows a sample IQA browser UI that allows users to preview the effectiveness of the tool by loading in an image of choice, looking at the *subjects* Clairvoyance was able to identify, as well as the predicted quality parameters. The IQA results are shown in Figures 10, 11, and 12, no over/under-fitting is observed. It can be seen that for good/bad predictions based on a given tolerance, the model achieves ~70% accuracy on validation. For magnitude of <10%, however, the model achieves only ~40% accuracy. This suggests that the IQA does reasonably well for selecting photos, but the magnitude of the predictions cannot be taken precisely. Interestingly enough, the sample images through the IQA browser did show relatively accordance for quality parameter values across the same image. Further, looking at the confusion matrix on a test-set, the overall accuracy for good/bad selection was almost 80%, with only 5.8% false-negatives (throwing away good images as bad). This is rather desirable, because false-positives can be verified by users after the selection has happened, whereas users are unlikely to go look through photos which have not been selected.

CycleGAN

When training a CycleGAN model, especially one for super-resolution, minimizing the loss functions does not necessarily equate to improving the model. The metrics of what makes an image better are not given to the CycleGAN. Instead, the discriminator loss and cycle-consistency are used to guide the training of the model. As such, instead of training the model for a large number of epochs, a model is saved for every 5 epochs. Then, the models were tested and qualitatively judged to select the best one.

Since the model was merely given two groups of images to learn a domain transform, to prevent the model from physically altering facial features, both the low and high quality images maintained the same variances in physical appearance by containing a variety of ethnicities and genders. Some results show amazing generated “quality” for focus, noise, and lighting:

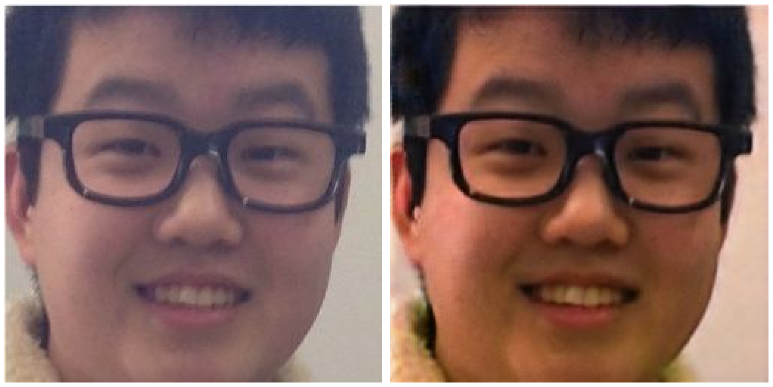


Figure 13: CycleGAN model trained to correct focus.

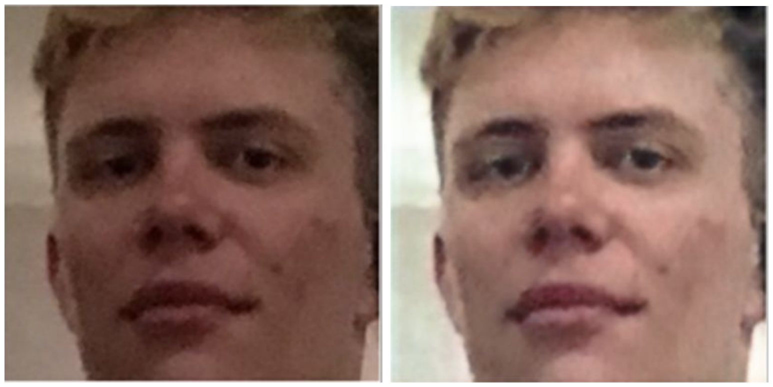


Figure 14: CycleGAN model trained to correct noise.

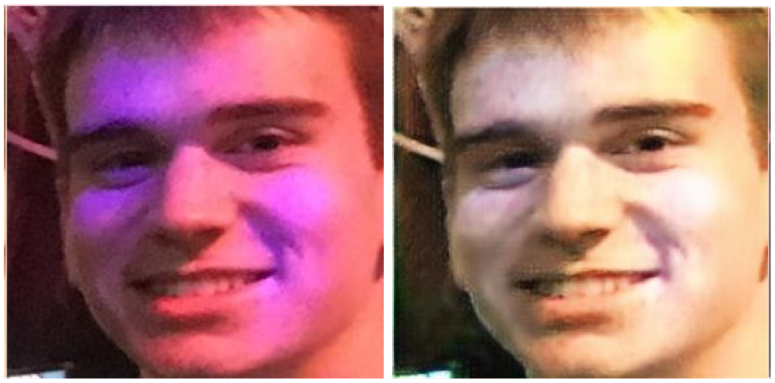


Figure 15: CycleGAN model trained to correct lighting.

However, the CycleGAN models were not without its flaws, some bad examples are shown:

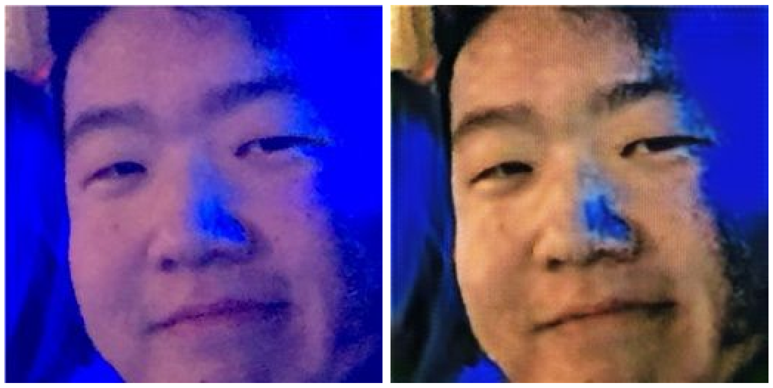


Figure 16: CycleGAN model causes discoloration of face.

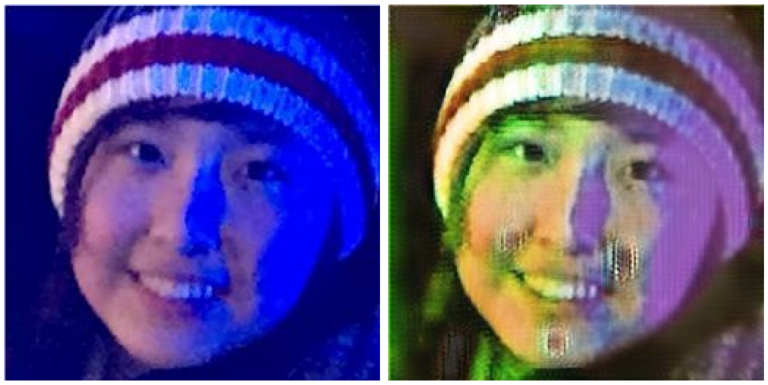


Figure 17: CycleGAN introduces artifacts in its output.

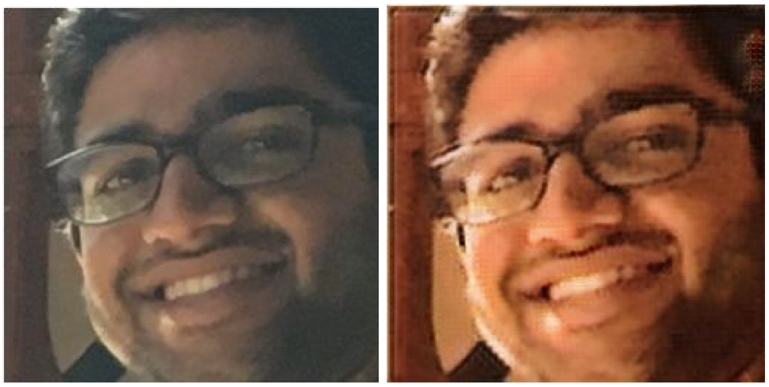


Figure 18: CycleGAN model causes image to be more pixelated.

Ethical Issues

The positive/negative effects of this form of post-processing automation on the photography industry is debatable. It would devalue a large portion of a professional photographer's work in selecting and editing photos. One can argue that it saves time and money, but may also lead to reduced employment of photographers and photo-editors to perform these automate-able tasks.

Further, this system may perpetuate certain biases regarding the *goodness* of a photo. If not correctly tuned, it could lead to photos with consistent high/low bokeh, contrast, brightness, etc. As we are assigning labels based on our own evaluation of the photos, bias is difficult to avoid, so the calibration details and our interpretations of the definitions are published here.

Key Learnings

Clairvoyance made apparent some of the difficulties associated with applying machine learning to non-standard sample tasks, like using CycleGAN to improve image quality instead of a style transfer from apples to oranges. It was found that it either generated amazing results that cannot be matched by photoshop/lightroom, or messed up completely. Also, forming datasets for training ML models is very difficult and time-consuming, and probably equally as important as the network architectures themselves.

Clairvoyance also exposed the good qualities of programming methods like OOP, which improved organization and readability of code, and enabled the use of “black-boxes” for easier collaboration. The Git version control system also contributed to the success of this project.

In the future, the authors hope to improve Clairvoyance with a GUI, and to work on merging generated results into original images, so that it can pass the IQA selection. For now, the one-click auto selection brings, hopefully, something new to the photography industry.

References

[1] Appendix A

[2] “Cascade Classification — OpenCV 2.4.13.7 documentation", *Docs.opencv.org*, 2018. [Online]. Available: https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html.

[3] Appendix B

[4] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", *Arxiv.org*, 2018. [Online]. Available: https://arxiv.org/abs/1505.04597.

[5] "CS231n Convolutional Neural Networks for Visual Recognition", *Cs231n.github.io*, 2018. [Online]. Available: http://cs231n.github.io/neural-networks-1/.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Cs.toronto.edu*, 2018. [Online]. Available: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf.

[7] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In International Conference on Computer Vision (ICCV), 2017.

[8] U. Demir and G. Unal. Patch-Based Image Inpainting with Generative Adversarial Networks. arXiv preprint arXiv:1803.07422v1, 2018.

[9] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", *Arxiv.org*, 2018. [Online]. Available: https://arxiv.org/abs/1412.6980.

Appendix A (Quality Parameter Definitions):

1. Poor focus means the subject is blurry/unclear, background blur (aka Bokeh) is fine.
2. Noisy means there are random variations in pixel brightness and colours in the subject's capture.
3. Poor lighting means subject too bright/dark, background lighting is ignored.

Appendix B (Examples of difficult situations for image evaluation):
Example 1: In photography (especially portrait photography), the background is often blurred to place the subject in more exaggerated focus. This results from using a larger aperture and hence offer increased depth of field (DOF). Thus, the photo may be mostly *unfocused* since the background takes up a large percentage of the photo area, but the photo is still considered *good*.

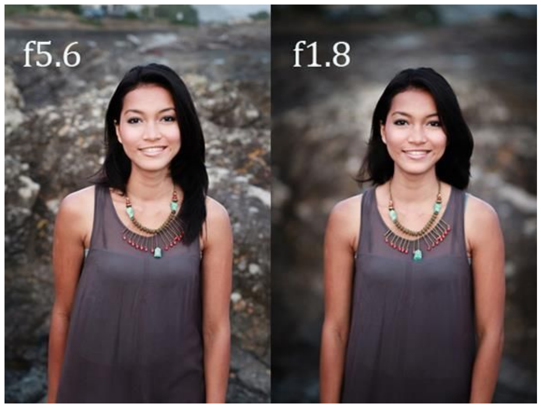


Illustration of *example 1*

Example 2: Then for lighting, the same scenario can happen with night versus day time lighting, indoor vs outdoor lighting. Even tougher, is to learn the different effects of lighting on people with different skin colours. Lighting, like focus, is a relative concept in photography, and requires analyzing the entire context of the image to classify.

Illustration of *example 2*