FACIAL IMAGE SUPER RESOLUTION FOR CARICATURIZATION

Lucas Best, Alexander Issa

The University of Texas at Austin, Department of Electrical and Computer Engineering, Cockrell School of Engineering, via 301 E Dean Keeton St, Austin, TX 78712

ABSTRACT

This paper presents a detailed application of using Super Resolution (SR) with Enhanced Deep Residual Networks (EDSR) on Low Resolution (LR) faces in order to acquire and manipulate matching High Resolution (HR) faces. As a differentiating factor for this project, this project's SR model was trained on a large dataset of HR and LR face image pairs. There are many developing applications of SR, and the goal of this project was to branch into the area of face SR in a fun way; through using SR face images to make a caricature of selected people in an image. The code used for this project can be accessed on Github.

Index Terms— Enhanced Deep Super Resolution, FFHQ Dataset, INRIA Person Dataset

1. INTRODUCTION

Image super resolution (SR) is a very popular category in the image processing field. While SR has many important applications in radar and sonar, such as MRIs and tomography, as well as other fields, it is not often used for more casual use cases [1]. The purpose of this paper is to introduce the idea of using SR for caricaturization, meaning to create images that look like sketches with enlarged faces. To the best of our knowledge, this has not been done before.

This paper assumes a background in traditional image processing techniques, while also trying to be thorough in explanations of the main techniques used. Specifically, the paper begins by delving into a brief explanation of convolutional neural networks (CNNs) evolving into Enhanced Deep Super Resolution (EDSR), and why this was the technology used. From there, background is given on the dataset acquired to train the model for the caricature process, then the model is evaluated using traditional machine learning and image processing techniques.

The paper then delves into the caricaturization application, explaining the step-by-step process used to turn an ordinary portrait photo into a caricature. Multiple examples are given of success, then limitations of the current implementation are discussed, namely with blending the SR image onto the original photo. Finally, the paper expands on future goals and ideas the project can take to expand upon these current results.

2. GROUP PROJECTS DURING COVID-19

We wanted to briefly acknowledge the development of this partner project during the COVID-19 pandemic.We have worked together on other projects and have known each other for many years. As a result, we understand each other's working styles, and know how to balance workloads. We worked together on every aspect of the project via Zoom calls, which we found was an effective way to get work done in parallel and stay on track. Lucas took primary responsibility over researching related work and code bases, finding datasets, and generating low resolution photos. Alex researched ways to accomplish the neural network goals of the project, wrote code for processing the data, training the models, and generating image results and information on the model itself. Both Alex and Lucas coded together to develop the final application notebook and worked together to develop the presentation video and this paper. We believe we split tasks and work time evenly, and we are pleased by the results of the project. Thank you for an awesome semester, Dr. Bovik!

Fig. 1. Example of a traditional caricature sketch [2].

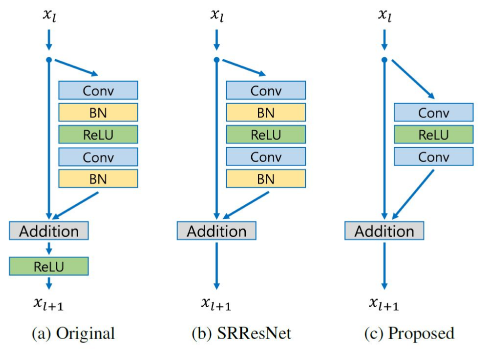

Fig. 2. Residual block designs from left to right: ResNet, SRResNet design, and EDSR [3].

3. BACKGROUND: CNN EVOLUTION TO EDSR

Given the primary goal of this project was to caricaturize a photo, the main distinguishable feature of a traditional caricature sketch is that the head of a person is abnormally large with exaggerated features when compared to the rest of the photo, as we can see in Fig. 1.

We created this effect using SR because the topic of trying to intelligently create a new, larger face from the original image seemed much more interesting than trying to grow the size using traditional upsampling methodologies like Nearest Neighbour Interpolation or Bilinear Interpolation. Specifically, the team used EDSR for this generation [3].

Before explaining EDSR, let's look at how it came to be. Deep CNNs are a fascinating technology in the image processing world, in that theoretically, a large enough network with enough time and data can accurately model any function and thus map an image to an output. While time and data can be bottlenecks, a more prevalent bottleneck suffered by former top-of-the-line networks like AlexNet, VGG-19, and GoogleNet was the "vanishing gradient" problem [4].

The vanishing gradient problem is that with more layers in the network, the gradient is back-propagated so many times that it becomes very small, practically zero [4]. This causes the network to stop in convergence or even degrade in performance, wasting all the additional time and computation spent to try to train a deeper network [4].

Residual Networks were a significant breakthrough, because they came up with the "identity shortcut connection" concept, which enables skipping one or more layers of the network [4, 5]. This enabled much deeper networks to be built that also tended to converge faster. There is one major caveat, which is that the given training dataset has to lead to a deterministic outcome, as these network designs are built on non-convex optimization, and will not converge otherwise [4, 5].

EDSR networks are a variation of the SRResNet architecture and won the 2017 NTIRE Super Resolution Challenge. As shown in Fig. 2, the main reason EDSR networks are a popular standard for SR is because they simplify the residual block significantly compared to original ResNet and SRResNet [5]. They remove the batch normalization layers, as well as the ReLU activation after the residual block. EDSR also has other optimizations, like scaling feature channels over layers with 0.1 scaling factor on the residual path, and many other optimizations [4, 5]. These design decisions result in using about 40% less GPU memory when training compared to SRResNet, which is nice given our lack of access to high performance GPU hardware [5].

4. BACKGROUND: THE DATASET

The dataset we used to train the aforementioned EDSR network consists of 5500 images of faces selected out of the Flickr-Faces-HQ (FFHQ) dataset [6]. As shown in the examples in Fig. 3, the FFHQ dataset consists of 70,000 1024x1024 HQ images of faces, and we utilized the first 5500 images from the Thumbnails subset, which had an image resolution of 128x128.

Fig. 3. Examples of the images in the FFHQ dataset in various resolutions. The picture in the leftmost column is 1024x1024 while the two rightmost columns have 128x128 sample images used for this project [4]. Image is rescaled for this paper.

4.1. Manipulation of the Dataset

The first step to getting a working dataset was to create HR and LR image pairs to train the EDSR model on. Since the goal of this project was to super resolve very far away and indistinct faces, the 128x128 set of 5500 facial images was treated as the HR set. In order to create a matching LR set, all images in the HR set were run through Pillow's "resize" function in order to downscale them to 32x32 [7].

The EDSR implementation base code we built our project on uses a pre-formatted dataset called DIV2K, which has constructors and methods built into it for creating image pairs and passing them through the EDSR model [8, 9]. Given that none of the available images in the DIV2K dataset met the needs of this project, the previously discussed FFHQ 128x128 to 32x32 facial image pairs dataset was created. In order to get this new dataset into a usable form, sklearn's "train-test-split" function was used to divide up the dataset into LR training, LR testing, HR training, and HR testing subsets [10]. The data was split such that 70% of it was dedicated for training and 30% was used for testing. Each of these divided sets was then converted into tensors in order for them to be usable inputs for Tensorflow Keras's model fit function [8].

5. EDSR TRAINING BREAKDOWN

Given the information presented on EDSR and the FFHQ facial image dataset, this section explains how the model was architected and trained, as well as the metrics used to evaluate and stop training. Visual results are shown in Sections 6 and 7.

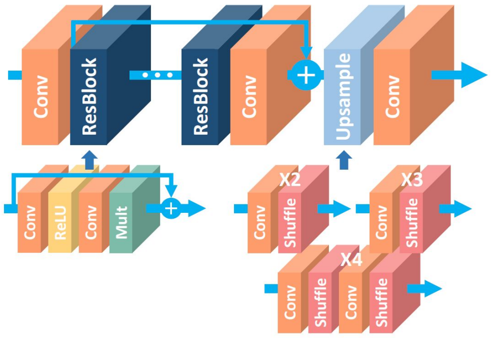

Fig. 4. Original EDSR network architecture [3].

5.1. Model Architecture and Parameters

The model design used here was similar to the proposed EDSR model and was specifically implemented by krasserm on Github using Tensorflow 2.x with Keras APIs [3, 8]. The architecture follows Fig. 4, with the main parameters being that our network had sixteen residual blocks/layers and sixty-four feature channels, as well as 4x upsampling. This is a smaller architecture than what the original EDSR publication used for the challenging 4x upsample problem, mainly because of a lack of access to powerful hardware and to further help prevent overfitting. Specifically, this network was trained using Google Colab's free GPU Python notebooks [11]. Additionally, this model used an Adam optimizer with piecewise constant decay of 1e-4 for 10000 steps, then all future steps at 5e-5. The loss function used was the mean absolute error (MAE). Images were shuffled each epoch, and the images were colored when trained to preserve post-processing flexibility in the caricaturization implementation.

5.2. Model Testing Numerical Results

In terms of results, the team terminated training after 40 epochs. This is because the team saw signs of convergence in the MAE of the training and test sets, as shown in Fig. 5. The team decided to stop at 40

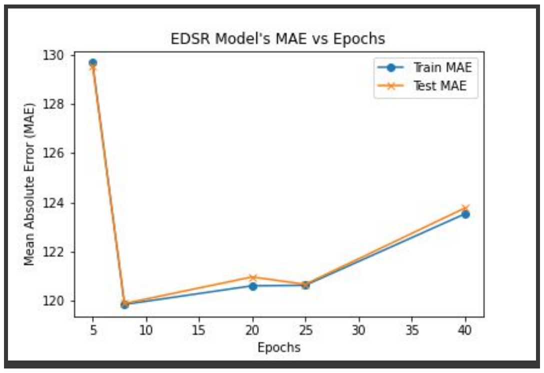

Fig. 5. Convergence of EDSR model shown via MAE at certain epoch iterations of training.

epochs to avoid further regression in the MAE, to further avoid overfitting, and also due to the aforementioned hardware constraints.

The final evaluation of the model on the testing dataset yielded a Peak Signal-to-Noise Ratio (PSNR) of 23.6367dB, and a Structural Similarity Index Measure (SSIM) of 0.8124. PSNR is effectively the ratio of the original image seen in the super resolved image, in comparison to the amount of noise in the super resolved image [12]. Due to the range of potential signals, PSNR is represented in decibels, and overall, higher is better. For context, the PSNR from the best 4x EDSR model in the original paper was 29.25dB [3]. SSIM is a perceptual-based method for determining the similarity of two images. SSIM uses structure, luminance, and contrast to factor structural interdependencies of photos when trying to score them [13]. 1.0 means the two images are the same. For context, the SSIM from the best 4x EDSR model in the original paper was 0.9017 [3]. Given these comparisons and the convergence, the team decided to move on to the application.

6. APPLICATION ON PORTRAIT PHOTOS

To show off the functionality of the trained EDSR model, select images from the INRIA Person Dataset were picked out with the intent of super resolving selected faces in the images [14]. Once said faces were super resolved and placed over the original image, a pencil-sketch effect was applied to the image to give the effect of a caricature. We will use Fig. 6 as a walkthrough example of how this process worked.

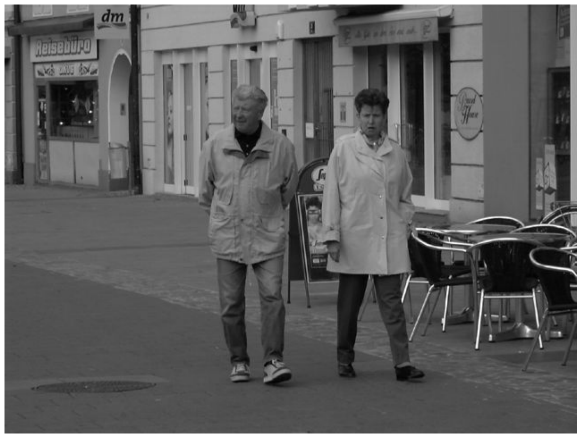

Fig. 6. An example image from the INRIA Person Dataset that was run through the caricaturization process [14].

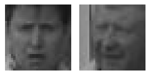

Fig. 7. Faces detected after running the face detection algorithm on Fig. 6.

6.1. Face Detection

Detecting faces and creating square facial image objects to run through the EDSR model was the first step in this process [15]. This was done through utilizing OpenCV's CascadeClassifier functionalities to detect faces, such as those in Fig. 7 [16]. The cascades provided by OpenCV use a combination of filters to detect Haar-like features on subsets of the image, eventually finding subsets that match all the cascading filters, leading to that subsection being classified as the goal object. For the purpose of this project, that goal object is a human face, so haarcascade_frontalface_default.xml was the applicable cascade to utilise [17].

Of the parameters passed to this function, three are of importance: scaleFactor compensates for faces that are further away from the camera, minNeighbors decides how many features must be detected to declare a face, and minSize affects the minimum window size that traverses the image to detect faces [16]. These parameters sometimes had to be adjusted based on the specific image the algorithm was run on. Once faces were found, the OpenCV function returned the top left corner coordinates,

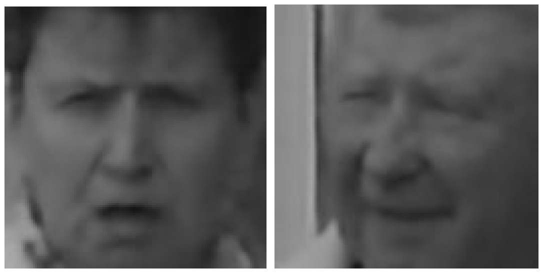

Fig. 8. The super resolved versions of the facial images from Fig.7.

width, and height, creating a square around each detected face. These variables were used to obtain the facial images to pass through the EDSR model.

6.2. Generating Super Resolved Faces with EDSR

All faces selected out of the original image were then run through the EDSR model, resulting in new super resolved facial images four times the original face's size. Examples of the super resolved faces from Fig. 7 are displayed in Fig. 8. The EDSR model does an excellent job of generating the super resolved faces, as clear facial reconstruction is done along with decent image clarity, and there are no signs of blocking artifacts, loss of major features, or overfitting.

6.3. Imposing SR Faces onto the Original Photo

The 4x super resolved facial images were then reimposed over the original image to give any selected persons the appearance of a 4x larger head [18]. We can see this result in Fig. 9. In order to get the appearance correct, the new facial image was transposed back onto each person's body by shifting the X-value left by 1.5x the original width and moving the Y-value up by 3x the height of the original facial image.

6.4. Applying a Pencil-Sketched Effect to Image

The last step of the process was to apply filters to the new image in order to give it a pencil-sketched effect [19]. First, the new image was converted into a grayscale image to be further manipulated. Then, a copy of the grayscale image, or mask, had a

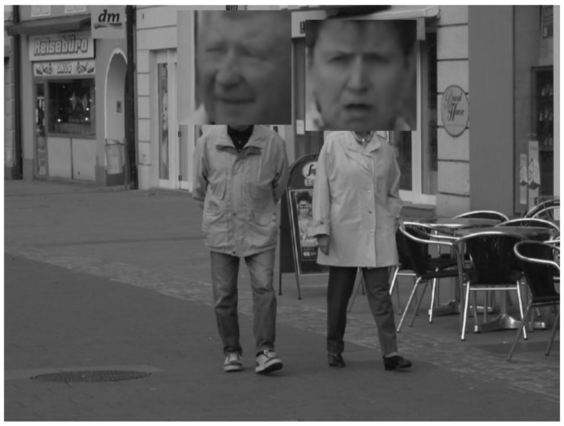

Fig. 9. Super resolved facial images from Fig. 8 transposed back onto the original image.

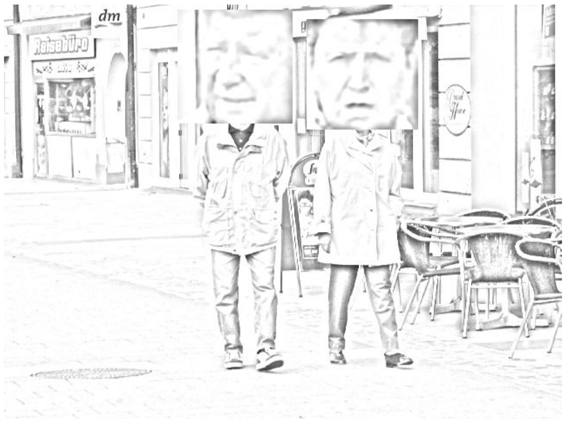

Fig. 10. The result of applying the pencil-sketching filters to Fig. 9.

bitwise-not applied to it, which emphasized the brighter and darker components of the image, bringing out the edges. This mask was then then gaussian blurred, or passed through a type of low pass filter, leaving mainly the edges of the photo. To get the final image, the original grayscale image was divided by the inversion of the masked image, or dodgev2, to give the appearance of a pencil-sketch. We can see the pencil sketch effect in Fig 10. This occurs because the mask takes everything except the original photo's edges to white, leaving the only gray edges at varying levels of intensity, which looks like a pencil sketch.

7. RESULTS

This section further recaps the information shown in

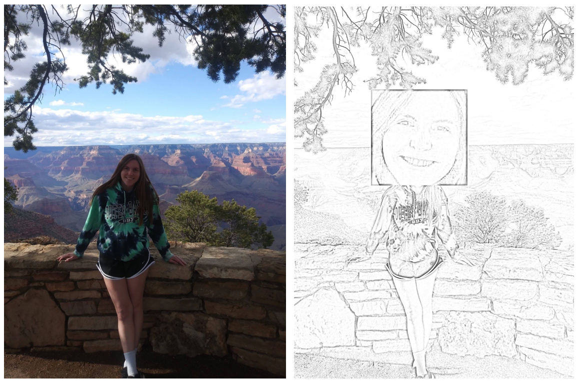

Fig. 11. Successful single-person detection, super resolution, and caricaturization. Courtesy to Selene Cullen for the photo.

Section 6, while showing limitations and failures that were also discovered during the development of this project.

7.1. Successful Caricaturization Examples

To build on Section 6, let's look at some more successful examples of caricaturization. In Fig. 11 and 12, we can see successful examples of caricaturization. These examples are deemed as successful because the SR faces generated are of high quality, the SR faces fit on the page, and they do not clash with each other or impair the background. Finally, the sketch effect looks very realistic in these photos.

7.2 EDSR Model Limitations

Technically, the EDSR model can take any input size of facial image. With that said, this is a bad practice, since as the input size increases, more data must be generated, and the model might not necessarily know what to do with the additional data provided, and thus predict inaccurate and non deterministic components in the generated image, or noise. For example, we can see in Fig. 13 that on the right side image, there are dotted pixel malformations in the prediction by the ear, forehead, nose, lips, chin, etc. when compared to the original image on the left.

Both images are scaled for visibility, but the original image size was 129x129, meaning the new image size was 516x516. This means the low resolution input passed to the model was significantly bigger than the 32x32 low resolution training set. We attribute these issues to that, combined with possible incomplete convergence of the model, noise in the dataset, possible lack of training photos where the face is at a slight angle to the camera, and so on.

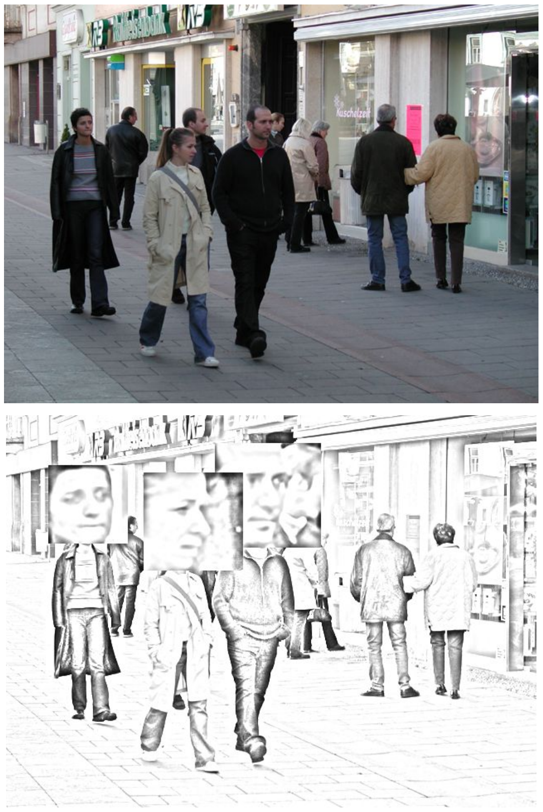

Fig. 12. Successful multi-person detection, super resolution, and caricaturization [14].

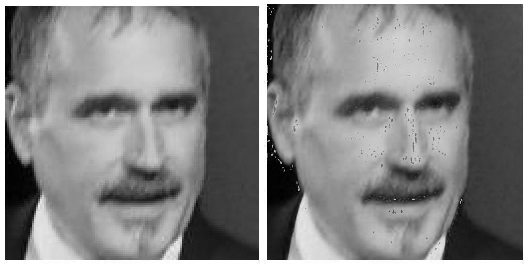

Fig. 13. Left image is the scaled up original face from a different photo [20]. Right side is the scaled down super resolved image, with visible noise in the generation.

7.3 Unsuccessful Edge Blurring Examples

There is one main issue that was not resolved in this project, which is how to effectively blend the SR image back into the photo. As Fig. 14 shows, gradient Gaussian mask blurring a rectangular box around the edges of the SR photo does not help blend it back into the photo [21]. If anything, we subjectively feel it makes the photo less appealing.

While we do not have time to implement this for the project, some proposed solutions are to either crop the training dataset images to circles/ovals of just the heads, and/or use circular facial detection, not rectangular. By cutting the background noise and solely writing over it with the circular SR head, the result of the photo should look more natural. Both options pose challenges, either with the dataset modifications, or accurately drawing a tight circle around the SR photo's head to crop it without important data loss; a detection problem of its own.

8. CONCLUSIONS

In this paper, a fun application of SR was proposed that demonstrated SR's ability to accurately output super resolved HR facial images when trained on a dataset of only facial images from FFHQ [6]. This application was tested on a variety of images and sizes, even outside of the INRIA dataset, although some images required manual adjustment of facial detection parameters, a feature that can possibly be automated [14].

There are many possible extensions to this project, with the main discussed one being to move to a circular face detection shape to cut irrelevant information from the current rectangular shape, leading to better super resolved head placement on the original image. Other ideas are to do colorized sketches, choose facial features to exaggerate, pick other objects to super resolve, change the weather, and change the background location. The hope of this paper is to inspire others to try training an SR model using a distinct dataset and create more entertaining applications of SR.

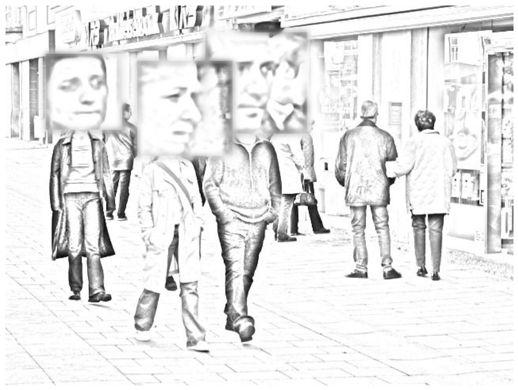

Fig. 14. Unsuccessful Gaussian blurring on edges of SR photo [14].

9. REFERENCES

[1] Linwei Yue, Huanfeng Shen, Jie Li, Qiangqiang Yuan, Hongyan Zhang, Liangpei Zhang, Image super-resolution: The techniques, applications, and future, Signal Processing, Volume 128, 2016, Pages 389-408, ISSN 0165-1684, https://doi.org/10.1016/j.sigpro.2016.05.002.

[2] Kolonja, Ermir. "Arnold Schwarzenegger - Pencil Caricature." *ArtStation*, 2017, Ermir Kolonja.

[3] B. Lim, S. Son, H. Kim, S. Nah and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Honolulu, HI, 2017, pp. 1132-1140, doi: 10.1109/CVPRW.2017.151.

[4] Fung, Vincent. "An Overview of ResNet and Its Variants." *Medium*, Towards Data Science, 17 July 2017, towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035.

[5] Tsang, Sik-Ho. "Review: EDSR & MDSR - Enhanced Deep Residual Networks for Single Image Super-Resolution (Super..." *Medium*, Medium, 27 Apr. 2020,

sh-tsang.medium.com/review-edsr-mdsr-enhanced-deep-residual-networks-for-single-image-super-resolution-super-4364f3b7f86f.

[6] NVlabs, NVLabs. “NVlabs/Ffhq-Dataset.” *GitHub*, 2018, github.com/NVlabs/ffhq-dataset.

[7] Pillow. “2.7.0.” *2.7.0 - Pillow (PIL Fork) 3.0.0 Documentation*, 2015, pillow.readthedocs.io/en/3.0.x/releasenotes/2.7.0.html.

[8] Krasser, Martin. “Krasserm/Super-Resolution.” *GitHub*, 2018, github.com/krasserm/super-resolution.

[9] Timofte, Radu. “DIVerse 2K Resolution High Quality Images as Used for the Challenges @ NTIRE (CVPR 2017 and CVPR 2018) and @ PIRM (ECCV 2018).” *DIV2K Dataset*, 2018, data.vision.ee.ethz.ch/cvl/DIV2K/.

[10] Scikit-learn developers. “Sklearn.model_selection.train_test_split.” *Scikit*, 2020, scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[11] Tensorflow Authors. “Introduction to Colab and Python.” *Google*, Google, 2018, colab.research.google.com/github/tensorflow/examples/blob/master/courses/udacity_intro_to_tensorflow_for_deep_learning/l01c01_introduction_to_colab_and_python.ipynb.

[12] “Peak Signal-to-Noise Ratio as an Image Quality Metric.” *NI*, 5 Mar. 2019, www.ni.com/en-us/innovations/white-papers/11/peak-signal-to-noise-ratio-as-an-image-quality-metric.html.

[13] Wang, Zhou, et al. “The SSIM Index for Image Quality Assessment.” *NYU CNS*, www.cns.nyu.edu/~lcv/ssim/.

[14] “INRIA Person Dataset.” *INRIA Person Dataset*, CVPR, 2005, pascal.inrialpes.fr/data/human/.

[15] Tiwari, Shantnu. “Face Recognition with Python, in Under 25 Lines of Code.” *Real Python*, Real Python, 5 Mar. 2019, realpython.com/face-recognition-with-python/

[16] “Cascade Classifier.” *OpenCV*, Google, 2020, docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.

[17] Opencv, vpisarev. “Opencv/Opencv.” *GitHub*, 19 Dec. 2013, github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml.

[18] “Using OpenCV to Overlay Transparent Image onto Another Image.” *Stack Overflow*, 2016, stackoverflow.com/questions/40895785/using-opencv-to-overlay-transparent-image-onto-another-image?fbclid=IwAR2FxAinlVh8GOKCZmQW0kZMOmAEt_caonwHTHPI3cDAiT3Yo-tJGTOmufs.

[19] Varma, Prudhvi. “Converting Image Into A Pencil Sketch In Python.” *Analytics India Magazine*, 10 Oct. 2020, analyticsindiamag.com/converting-image-into-a-pencil-sketch-in-python/.

[20] Bovik, Alan. “Professor Alan Bovik's Emmy Acceptance Speech.” *YouTube*, 5 Nov. 2015, www.youtube.com/watch?v=q1BbIVlwtkY&ab_channel=WirelessNetworkingandCommunicationsGroup.

[21] “Gradient Mask Blending in Opencv Python.” *Stack Overflow*, Sept. 2017, stackoverflow.com/questions/42594993/gradient-mask-blending-in-opencv-python.