Image Enhancer using Super-Resolution, Edge Detection and Morphological Operations

Project report for IITB EE610 Image Processing 2021

Shreyas Nadkarni
Dept of Electrical Engineering
IIT Bombay
Mumbai, India
19D170029@iitb.ac.in

Tushar Nandy
Dept of Electrical Engineering
IIT Bombay
Mumbai, India
190020125@iitb.ac.in

Vinamra Baghel
Dept of Electrical Engineering
IIT Bombay
Mumbai, India
190010070@iitb.ac.in

***Abstract*—This project has been done as a part of the course EE610: Image Processing (Autumn 2021 offering), under Prof. Amit Sethi, Dept of Electrical Engineering, IIT Bombay. An application for editing and enhancing an image based on edge detection algorithms, morphological operations and a Deep-Learning based super-resolution approach has been designed. This report gives a lucid explanation of the background and methodology behind the work, the experimental observations using the tool, and an overview of this project.**

I. Introduction

In the digital era of computers and smart phones, images are seen everyday. Image enhancement is a set of techniques applied to an image to emphasize certain aspects of an image for better understanding. The techniques may be applied to whole or a part of the image. These techniques are particularly useful in forensic and police investigation, pathology, medical diagnosis, satellite imaging and data extraction as well as in exploration of marine resources.

Edge detection is a set of mathematical operations performed on an image to highlight borders between segments of distinct contrast and color. There is often a sharp change of values across these edges. This method is motivated by the 1-dimensional "step-detection" in signals. A common approach includes calculating the gradient across all the three channels of the colored image and applying heuristic-based methods. This is an important step in image segmentation and depth-estimation.

Morphological operations are used to detect and emphasize the presence of particular shapes and frames from an image and are often used to de-noise an image before clustering.

Super-resolution is a set of techniques which aim to enhance the resolution of an image. Common techniques are based on the use of ML-based methods which interpolate the features from the low-resolution image. Zooming small images often results in pixelated images which often result in the lack of high frequency components. This method is particularly useful to enlarge images captured from cameras with limited resolutions.

This project has been done in three parts: edge detection, morphological operations, and super-resolution, and have been combined in a single GUI application for real-time use on images, with the functionality to browse and save images, as well as undo the transformations.

II. Background and Prior Work

A. Edge Detection

Edge detection has been tried using several methods in the past, with techniques such as Marr-Hildreth Algorithm and Canny Edge Detection Algorithm, apart from relatively less complex techniques such as gradient-based methods using Prewitt or Sobel operators. In this project, we have implemented two of these, namely the Canny Edge Detection algorithm and the simple thresholding gradient method of edge detection and the mathematics involved in the two techniques. The overview of these methods, with the mathematical details behind Canny's algorithm have been presented as follows.

1) Marr-Hildreth Algorithm: This is an alternative technique which has been tried in the past. It was designed by David Marr and Ellen Hildreth in 1980, the details of their experiments can be found in [1]. The technique involves computing the Laplacian of the Gaussian (LoG) kernel hence defining a new "LoG kernel" and convolving it with the input image. The zero crossings of the convolution result are then obtained to determine the location of edges in the original image. This was one of the earliest successful techniques in edge detection.

2) Canny Edge Detection Algorithm: This algorithm was designed by John Canny in 1986, the details of which can be found in [2]. Although the exact details of the algorithm may vary between multiple implementations, the general idea remains the same. The exact computational steps of the algorithm (coded from scratch using the NumPy and SciPy libraries of Python) followed in this project have been described here:

1) Image Smoothing: We convolve the image $f(x, y)$ with a Gaussian kernel to lessen sudden unwanted changes

in intensity between two pixels leading to a shoot in gradient values.

$$f_s(x,y) = G(x,y) * f(x,y) \tag{1}$$

where $f(x,y)$ takes the values of $R(x,y)$, $G(x,y)$ and $B(x,y)$ one after the other

2) Gradient Computation: Since we are working with color images, we need to incorporate the three channels R,G and B into the computation.

$$g_{xx} = |\frac{\partial R}{\partial x}|^2 + |\frac{\partial G}{\partial x}|^2 + |\frac{\partial B}{\partial x}|^2 \tag{2}$$

$$g_{yy} = |\frac{\partial R}{\partial y}|^2 + |\frac{\partial G}{\partial y}|^2 + |\frac{\partial B}{\partial y}|^2 \tag{3}$$

$$g_{xy} = |\frac{\partial R}{\partial x}||\frac{\partial R}{\partial y}| + |\frac{\partial G}{\partial x}||\frac{\partial G}{\partial y}| + |\frac{\partial B}{\partial x}||\frac{\partial B}{\partial y}| \tag{4}$$

$$\theta(x,y) = \frac{1}{2}tan^{-1}(\frac{2g_{xy}}{g_{xx} - g_{yy}}) \tag{5}$$

$$F_\theta(x,y) = \{\frac{1}{2}[g_{xx}+g_{yy}+(g_{xx}-g_{yy})\cos(2\theta(x,y))+ 2g_{xy}\sin(2\theta(x,y))]\}^{\frac{1}{2}} \tag{6}$$

where $\theta(x,y)$ is the direction of maximum rate of change of the vector $c(x,y) = [R(x,y) \quad G(x,y) \quad B(x,y)]^T$ and $F_\theta(x,y)$ is the rate of change of $c(x,y)$ in the direction $\theta(x,y)$.

3) Non-maxima suppression:
   - Depending on $\theta(x,y)$ assign a direction to the gradient for each pixel out of four possible directions: Vertical, Horizontal, $+45°$, $-45°$ as shown in Figure 1. Let this direction be d.
   - If the gradient at a particular pixel is less than one of both of the neighbors of the point (x,y) along direction d, assign $G_N(x,y) = 0$, otherwise assign $G_N(x,y)$ equal to the gradient value.

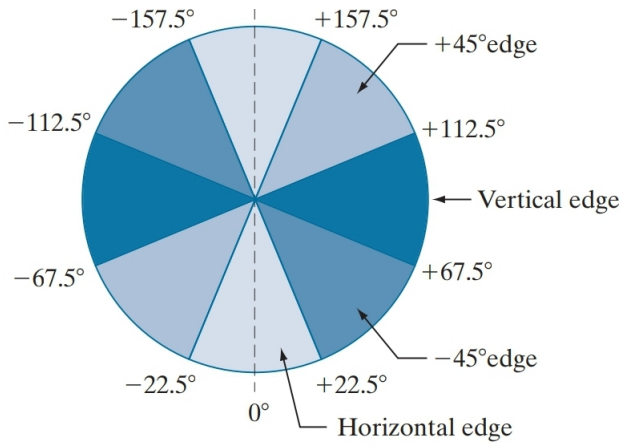

Fig. 1. The angle ranges depending on which the direction d is assigned. *(Reference: Page 731, Digital Image Processing by Gonzalez and Woods, 2002)*

4) Hysterisis Thresholding: This takes two parameters T_H and T_L which are the higher and lower threshold values. The pixel values are changed as follows:
   - If $G_N(x,y) \geq T_H$, assign value 255 to that pixel. This is then a "strong pixel" which means that it has a strong chance of being on an edge in the image.
   - If $T_H > G_N(x,y) \geq T_L$, assign some temporary value between 0 and 255 (say 20), to the pixel. This is then a "weak pixel".
5) Connectivity Analysis: This technique is to iterate over the image, and resolve the weak pixels into either a strong pixel (an edge point) or a non edge point. The method used here is 8-connectivity, wherein the new value assigned to a weak pixel becomes 255 if there is a strong pixel in the 3x3 box with this pixel at the centre, and 0 otherwise. Thus this final technique gives a set of points which correspond to edges in the original image, thus completing the algorithm.

3) Thresholding Gradient Method: This method is relatively simple compared to the Canny Edge Detection Algorithm. The first two steps are same as in the Canny Edge Detection Algorithm (smoothing and computing the gradient: equations 1 to 6). However, here we simple define a threshold value T and process the gradient $F_\theta(x,y)$ of the image. If the gradient magnitude is above the threshold T, a value of 255 is assigned, and if it is below the threshold, a value of 0 is assigned. In other words, it is a simple binary thresholding of the gradient image.

B. Morphological Operations

Morphology is a theory and a set of techniques which deals with the geometrical structures of various features in an image using the language of set theory. Various morphological operations such as erosion, dilation, opening, closing, convex hull, thinning, thickening, etc are used independently as well as in combinations to develop algorithms for shape-oriented tasks in images such as hole filling or boundary detection. In this project we have coded the two most basic of these operations namely dilation and erosion, the explanations of which are presented in this section. To understand them, we need to define two preliminaries: reflection of a set about its origin and translation of a set by a point z.

- The "reflection" of a set B about its origin is defined as $\hat{B} = \{w \mid w = -b, for \ b \in B\}$
- The "translation" of a set B by point $z = (z_1, z_2)$ is defined as $(B)_z = \{c \mid c = b + z, for \ b \in B\}$

Let A and B be two sets in ^{2}. Here, we want A to be our image (a 2-D array of pixel values) and B to be a "structuring element".

1) Dilation: The dilation of A by B is defined as $A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \phi\}$. It consists of all displacements z such that $\hat{B}$ overlaps with at least one element of A.
2) Erosion: The erosion of A by B is defined as $A \ominus B = \{z \mid (B)_z \subseteq A\}$. It consists of all displacements z such that $(B)_z$ is entirely contained in A.

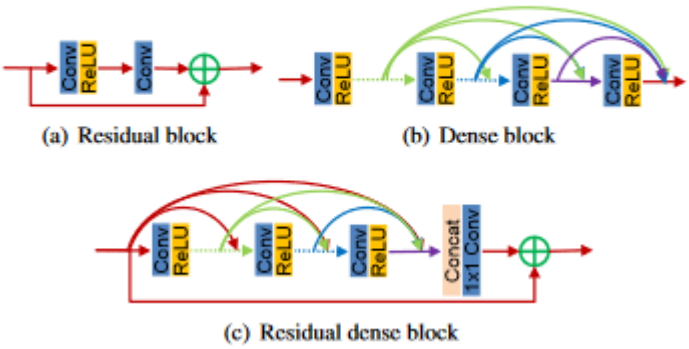

Fig. 2. Comparison of Blocks (Ref: [4])

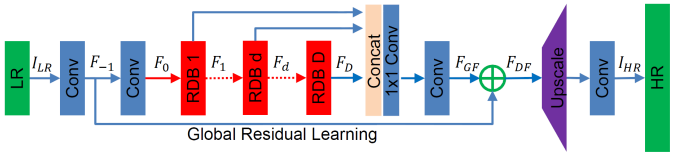

Fig. 3. Residual Dense Network (Ref: [4])

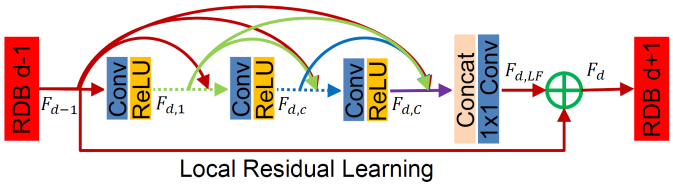

Fig. 4. Residual Dense Block (Ref: [4])

C. DL based Super-resolution

In deep neural networks, one often encounters the notorious problem of vanishing gradients which can inhibit learning and result in great deviations of results. The following three techniques are commonly used to tackle this issue.

1) Residual Learning in Neural Networks: Consider a set of neural network layers denoted by a function $f(\cdot)$. If $f_{res}(\cdot)$ was to denote a residual block, the its output is of the form

$$f_{res}(x) = f(x) + x$$

This addition of input and output is realised by appropriately padding 'x' such that x and $f(x)$ are of the same dimensions. The weights of the next layer act (or convolve) to the net sum of $f(x) + x$. This means

$$g(f_{res}(x)) = W_1 * (f(x) + x)$$

2) Skip-connections: A skip-connection is similar to a residual connection, except that the input is not added but concatenated to the output. In effect, if x is an input to a layer $f(\cdot)$, then the output of the next layer $g(\cdot)$ is

$$g(\cdot) = W_1 * f(x) + W_2 * x$$

Since the weights used for x and $f(x)$ are different, they needn't be of the same dimension.

3) Dense Block: A dense block is a set of layers in which the output of a layer is "skip-connected" to all subsequent layers. Fig 2 shows all the three blocks in comparison.

III. Data and Methodology

A. Edge Detection

The two edge detection techniques we implemented were Canny Edge Detection and Thresholding Gradient method. To compute the gradient in each we used the 3x3 Sobel operators for edge detection in the x and y directions separately. Convolving these operators with an image gives us the gradient directly for a grayscale image, while for a three-channel RGB image, we need to convolve them with each of the channels and use the formulae presented in the last section. After computing the gradient, the thresholding gradient method simply involves binary thresholding about a user defined threshold, whereas the Canny detection method involves additional processing for non-maxima suppression, hysterisis thresholding and connectivity analysis. For the implementation we primarily used the NumPy library, alongwith the "convolve2d" function from the "signal" module of the SciPy library.

B. Residual Dense Network for SR

The network architecture described in [4] is a 4-part collection of the following modules: shallow feature extraction, residual dense blocks, global feature fusion and up-sampling (See Fig 3). Input to each layer is padded appropriately such that the output is a scaled version of the input image.

1) Shallow Feature Extraction: This module consists of two convolution layers which learn high-level features from the image. The output of the first layer (F_{-1}) is used for global residual learning, while the output of second shallow layer (F_0) is used as input to the following residual blocks. The number of features learned by the shallow blocks is denoted as G_0.

2) Residual Blocks: The power house of this architecture is a residual dense block. This is a combination of the previously discussed blocks. Output from each layer is bypassed to all subsequent layers, and the output is then added to input for residual learning (see fig 3). The number of features learned by each layer in the block is called growth rate and is denoted by G. The number of convolution layers in the block is denoted as C. The final step, in which the residue is added to the input is called Local Feature Fusion. It is a "fusion" of the local features learned by the block and the input features to the block. This enables the following blocks/layers to fully utilize the local features from all previous blocks.

3) Global Feature Fusion: This form of skip-connection is applied across all the residual blocks by appending their outputs. The concatenation of residual-blocks is fed to a 1×1 convolution layer which converts the features from G to G_0. The output of layer F_{-1} is then added as residual connection.

4) Upsampling: The upsampling layer is required to upscale the coarse resolution features to interpolate and scale the final image by the scaling factor. The result of the upscaling layer is a 3-channel output with dimensions that n times that of the input image (n being the scaling factor).

IV. Experiments and Results

We have presented some sample images which we obtained by using our coded techniques on images. For edge detection, it is seen that for natural images, the Canny edge detection algorithm works better than the thresholding gradient method as in Figures 5 and 6, in the sense that it includes only the most relevant pixels in the set of edge pixels. However for images in which the intensity remains constant over a long range such as cartoon images, like Figure 11, it is seen that the thresholding gradient algorithm performs better. This might be attributed to the fact that the gradient if non zero does have significant magnitude in case of non-natural images and hence the thresholding algorithm succeeds in catching these pixels, while the Canny algorithm rejects a few of them as weak pixels and fails to give a continuous edge.

The output of the edge detectors can be dilated using the morphological dilation tool, in order to display the edges more prominently. Significant improvement is seen in the output of the Canny Edge detector after dilation.

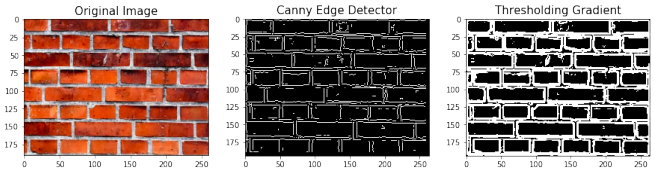

Fig. 5. An image of a brick wall and the corresponding outputs using Canny Edge Detector and Thresholding Gradient. The threshold values for Canny detector are Th = 50, Tl = 25, and that for the thresholding gradient method is 50.*(Image Ref: https://unsplash.com/photos/rhaS97NhnHg)*

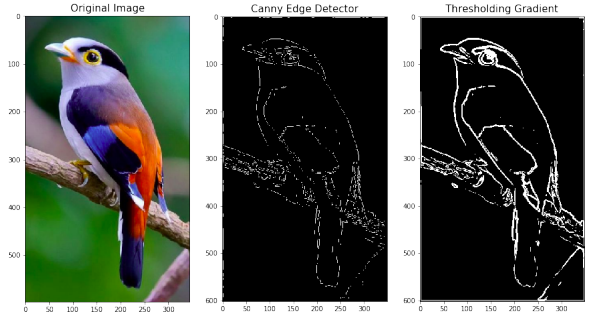

Fig. 6. An image of a bird and the corresponding outputs using Canny Edge Detector and Thresholding Gradient. The threshold values are same as in the previous image.
(Image Ref: https://in.pinterest.com/pin/19281104631420541/)

To demonstrate the use of the morphological operations independently, figures 9 and 10 have been presented. Erosion shrinks the white parts of the image while dilation thickens them, and vice versa for the black parts, as is expected from the definitions of these operations.

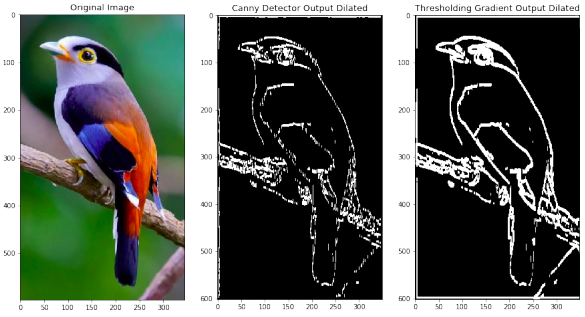

Fig. 7. The resulting images after applying dilation using a 5x5 square structuring element.*(Image Ref: https://in.pinterest.com/pin/19281104631420541/)*

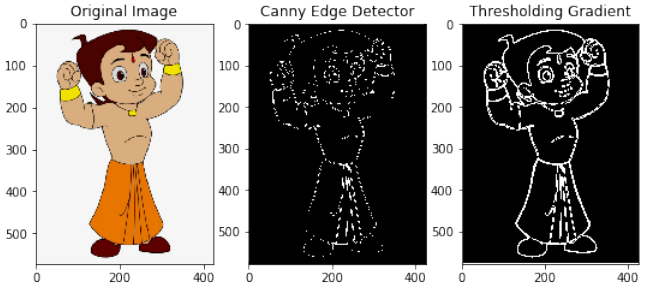

Fig. 8. An image of a cartoon on which the thresholding gradient method seems to perform better than Canny Edge Detection method. *(Image Ref: https://www.snapdeal.com/product/asian-paints-chhota-bheem-and/643904791025)*

The results of Super-resolution can be seen in Fig. 11. The central image dimensions were reduced 4 and then re-scaled using bicubic interpolation. The amount of blurring is a sign of the loss of information. Fig. 11 (c) is the result of passing the low-resolution image through the trained neural network. The text on the top of the yellow umbrellas has been considerably restored.

V. Learning, Conclusions, and Future Work

Thus, we have presented the theory behind and utility of edge detection, morphological erosion and dilation, and super-resolution. These techniques and their combinations provide interesting ways to process and enhance images, according to requirement.

This project was a great learning experience for us. We got an opportunity to implement some of the techniques we learnt theoretically, in a practical way. We got an overview of these and related techniques and implementing them helped us understand them in a better way.

With the advent of deep learning, future work in images is greatly focused on developing better models for accomplishing tasks such as image segmentation, object detection, etc. However, traditional methods still hold significance for complementing machine learning, for tasks such as pre-processing of images. Future work in edge detection can be towards

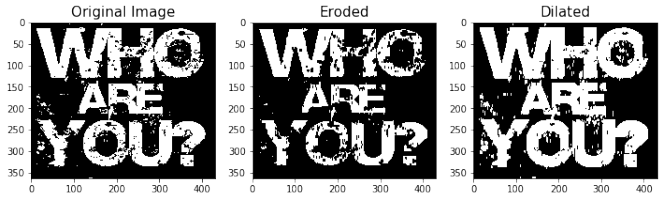

Fig. 9. The effect of morphological operations on the thickness on the white parts. *(Image Ref: https://www.wallpaperflare.com/white-text-on-black-background-typography-artwork-wallpaper-244149)*

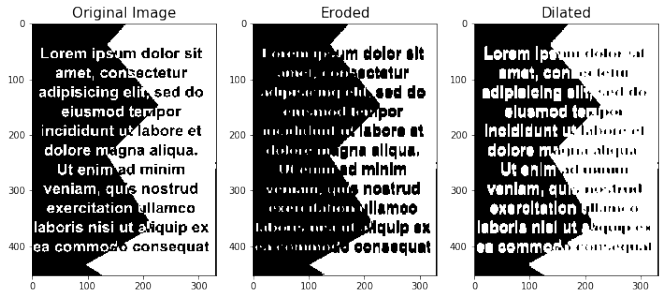

Fig. 10. Morphological operations on text. Erosion causes white parts to shrink while dilation does the opposite. *(Image Ref: https://stackoverflow.com/questions/7392585/how-to-display-text-with-two-color-background)*

developing more such algorithms, perhaps with the use of different kernels or modification in a particular step.

A major disadvantage of using the present neural network for super-resolution is the huge number of parameters. This is quite evident in the running time taken by our GUI to produce the scaled images. Another approach that can be used for super-resolution is the use of GANs. The load of learning is distributed between two sub-models called the generator and the discriminator. Finally, it is only the generator that is used in deployment. A sizeable learning point from this project, for our team, is our first review and reproduction of a neural network architecture. Prior to this, none of our members had coded a neural network solely on the basis of its understanding from its research paper. We participated in long discussions and collectively understood the architecture on our own.

Contribution of Team Members

Shreyas Nadkarni: Programmed the Canny Edge Detector and gradient computation, Erosion, Dilation and combined them into the GUI

Tushar Nandy: Reviewed papers. Coded the neural network for super-resolution, trained it, and combined the testing code of the super-resolution part with the GUI. Also made the project video.

Vinamra Baghel: Reviewed papers on Superresolution and edge detection, helped in coding the GUI and drafting the report.

Acknowledgements

We thank Prof. Amit Sethi, course instructor for EE610: Image Processing for giving us the opportunity to do this project and complement the theoretical knowledge gained through this course with some practical implementation of techniques. His guidance and the learning obtained from the course have been instrumental towards this project. We also thank the teaching assistants Gouranga Bala, Nikhil Cherian Kurian, Sachin Yadav, Anubhav Goel, Rajat Kumar Panigrahi, Abhishek Rajawat, Sanchi Mangulley, Ravi Kant Gupta and Pradumn Kumar for their support to us as well as other students in the course.

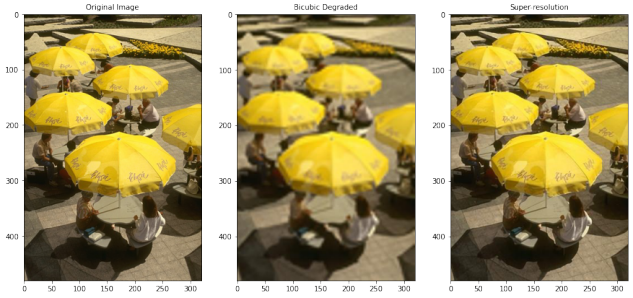

Fig. 11. (a) Original Image (b) Bicubic Degraded (c) Super-resolution *(Image Ref: [6])*

References

[1] Marr, D.; Hildreth, E. (29 Feb 1980), "Theory of Edge Detection", Proceedings of the Royal Society of London. Series B, Biological Sciences, 207 (1167): 187–217

[2] John Canny, "A Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: PAMI-8, Issue: 6, Nov. 1986)

[3] Digital Image Processing, Fourth Edition (2002), R. Gonzalez; R. Woods, Pearson Publications

[4] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, Yun Fu, "Residual Dense Network for Image Super-Resolution", CVPR, 2018

[5] Paszke et al, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", Advances in Neural Information Processing Systems 32, 8024–8035, 2019

[6] D. Martin and C. Fowlkes and D. Tal and J. Malik, "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics", Proc. 8th Int'l Conf. Computer Vision, 416–423, 2001

[7] Blogs and Online Sources:

- https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123
- https://stackoverflow.com/questions/57033158/how-to-save-images-with-the-save-button-on-the-tkinter-in-python
- https://www.geeksforgeeks.org/file-explorer-in-python-using-tkinter/
- https://stackoverflow.com/questions/10133856/how-to-add-an-image-in-tkinter
- https://github.com/yjn870/RDN-pytorch