Spatial resolution enhancement for halftone images using convolutional neural networks

Guilherme Apolinário Silva Novaes
Dept. Eng. Sistemas Eletrônicos, Escola Politécnica
Universidade de São Paulo
São Paulo, Brazil
g.novaes@usp.br

Hae Yong Kim
Dept. Eng. Sistemas Eletrônicos, Escola Politécnica
Universidade de São Paulo
São Paulo, Brazil
hae.kim@usp.br

***Abstract*—To print a grayscale image on a printer with current technology (laser or inkjet), it is necessary to convert it into a binary image called *halftone*, so that when a human being looks at it from a distance, the binary image visually resembles the original image. In this work, we present a technique to increase the spatial resolution of halftone images based on convolutional neural networks (CNNs). To our knowledge, this is the first work that seeks to solve this problem using CNNs. We use our algorithm to increase the resolution of binary halftones and show that our algorithm is considerably better the previous decision-tree based method. We also use our algorithm to increase the resolution of scanned old comics pages, and show that our algorithm is better than conventional resampling methods.**

***Index Terms*—Halftone, convolutional neural networks, deep learning, up sampling.**

I. Introduction

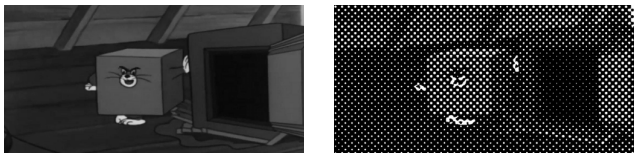
As laser and inkjet printers emulate continuous-tone images using tiny dots, it is necessary to use halftoning techniques during the pre-printing process to convert continuous-tone images to their binary versions so that the binary image visually resembles the original image when viewed by a human being from a certain distance.

In other words, given a grayscale image $G : \mathbb{Z} \to [0, 1]$, the halftoning process generates a binary image $B : \mathbb{Z} \to \{0, 1\}$ such that for any pixel p, $\bar{B}(p) \approx G(p)$, where $\bar{B}(p)$ is the mean value of the image B in a neighborhood around the pixel p.

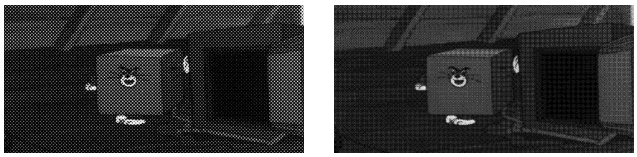
The spatial resolution of a halftone image to be printed by devices is related to two concepts:

- Dots per inch (DPI): DPI is the maximum number of tiny dots a printer can print per inch. Each tiny dot can only be off (white) or on (black), thus there are no shades of gray. DPI is a hardware-bound characteristic. For instance, computer screens, mouse optical devices, and printers are measured by DPI.
- Lines per inch (LPI): To achieve the appearance of grayscale on a printed sheet, the printer uses an optical illusion by printing rounded halftone dots of varying sizes and, by printing them at high resolution, gives the appearance of grayscale [3], [8]. Each rounded dot is made up of many tiny dots. A row of rounded dots is called a line. Thus, LPI is a measure of the number of rounded dots per inch. LPI is a feature linked to the software, as the halftone algorithms are responsible for creating these rounded dots. Figure 1 features a comparison of a grayscale image with its halftone representations at 75, 150, and 300 LPI.

This work was partially supported by the Coordination for the Improvement of Higher Education Personnel (CAPES), Brazil



(a) Original grayscale image (b) 75 LPI halftone image



(c) 150 LPI halftone image (d) 300 LPI halftone image

Fig. 1: Halftone of a grayscale image in different LPIs.

The literature on increasing the spatial resolution of binary halftone images is scarce. Kim [2] presents windowed zoom decision trees (WZDT) to solve the problem of increasing DPI and LPI together, achieving the optimal results when using a fixed small window, with up to 2% error when increasing the resolution of coarse dots, fine dots, and clustered dot ordered dithering, in pre-print images. A pre-print image is a binary halftone software-generated image that did not pass through print-scan process.

Our work extends the previous work [2], studying the problem for both pre-print (processed only by halftoning algorithms) and post-print (printed and re-scanned) halftone images. To achieve this goal, we use convolutional neural networks (CNNs) [5] to increase spatial resolutions of pre-

979-8-3503-2028-2/22/$31.00 ©2022 IEEE
DOI 10.1109/CSCI58124.2022.00052

print and post-print halftone images, without the need to have the original image in gray levels. To our knowledge, this is the first work that seeks to solve this problem using CNNs.

II. Upsampling of halftone images

A. Types of halftone

There are two main categories of halftone:

- *Clustered-dot or pattern halftone*: The rounded halftone dots are generated accordingly some geometric structure, distributed in a homogeneous way, where the variation of the dot size represents the level of the gray shades. Pattern halftones are usually created by ordered dithering halftone algorithms, and they are normally used in magazine and newspaper printings and in laser printers.
- *Dispersed-dot or error diffusion halftone*: The halftone dots are usually generated by error diffusion halftone algorithms. The dots are scattered in a heterogeneous way, thus creating more possibilities of generating gray tones. As this technique does not take into account the rounded dot patterns, there is no way to specify LPI in this technique. This technique is usually used in inkjet printers.

We can also speak of a third "intermediate" category, dispersed-dots ordered-dithering, where tiny dots are not organized as rounded halftone dots, but spatially dispersed following some pattern.

This work will only address the following pattern halftones: Euclidean, square, circle and triangle, represented in the figure 2. The error diffusion halftone algorithms have a chaotic behavior that makes it a very difficult task to increase the resolution by machine learning algorithms [2].

(a) Euclidean (b) Squared (c) Circle (d) Triangle

Fig. 2: Examples of pattern halftones.

B. About the problem

Let us consider a grayscale image I with $m \times n$ pixels, and its ideal equivalent D in high resolution, with double resolution $2m \times 2n$ pixels. To recreate the image D from I, it is necessary a function r, where $r(I) \approx D$. Evidently, there is no function r that can generate perfect D from I, since there are many different high-resolution images D that correspond to a low-resolution image I. In other words, this problem is ill-posed.

C. CNNs for up sampling

Dong et al. [5] present one of the first neural network architectures for up sampling images, and they argue that, for problems related to re-sampling, CNNs are a great method for preserving details.

The major differences between conventional re-sampling (such as nearest neighbor, bilinear or bicubic interpolations) in relation to CNN resampling is due to three properties [5, 9]:

- Patch extraction: input features are extracted by windowing operations from the convolution layers;
- Non-linear mapping: convolutions with different window sizes (kernels) allow more efficient patch extraction, leading to more effective learning;
- Reconstruction: through the non-linear mapping, it is possible to perform a high fidelity re-sampling in order to reconstruct the ideal image.

Frank et al. [11] introduces some machine learning techniques for halftoning, however these methods aim to convert a grayscale image into a high quality halftone image, resulting in error diffusion dispersed-dot halftone images.

Lim et al. [7] describe a specific CNN architecture to increase the resolution of color images, the enhanced deep super-resolution network (EDSR), one of the best current techniques for preserving image details. This architecture also reduces GPU memory usage by about 40% compared to other architectures with the same purpose, as this architecture does not use the batch normalization layers.

The EDSR architecture is based on the well-known ResNet, which uses residual blocks: an input I passes through a series of layers, such as convolution and activation layers, generating an output O. Finally, the output of this residual block is given by $I + O$, making the block learn O more easily.

Furthermore, as can be seen in the figure 3, the architecture has additional up sampling blocks, responsible for increasing the resolution of the input image. These blocks use a combination of the 2D convolution sequence with the *depth to space* layer (also known as *pixel shuffle*), needed to increase the spatial dimensions of the convolution output matrix. The proposed model alternates three types of output blocks for 2×, 3× and 4× up samplings. For this work, we consider only 2× up sampling.

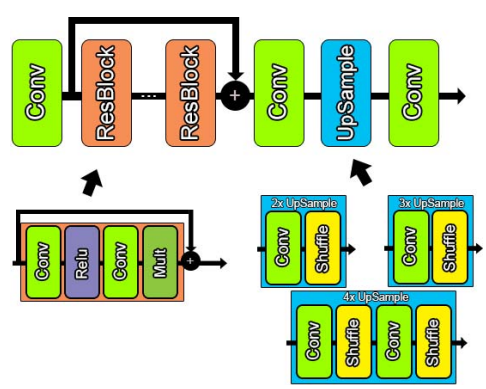


Fig. 3: EDSR architecture

III. Methodology

In order to carry out the experiment, we performed two steps:

1) Organization of the dataset: which consisted of the selection of images from the dataset for halftoning, based on the information provided by the literature and on the needs of the proposed problem;
2) Model training: which consisted of training the EDSR model, to compare the results with the WZDT algorithm [2]. For each dataset, similar training was performed using the EDSR and WZDT techniques, to make a fair comparison.

A. Dataset organization

In this work, we use two types of dataset, indicated as (A) and (B) (figure 4). The dataset of type (A) have only pre-print images, i.e., binary halftone images that were generated from a grayscale image by some algorithm (by the printer driver or by some function from a library). The type (B) dataset consists of post-print images, that is, images that have been printed on paper and re-transformed onto digital media (e.g., by scanning the printed images).

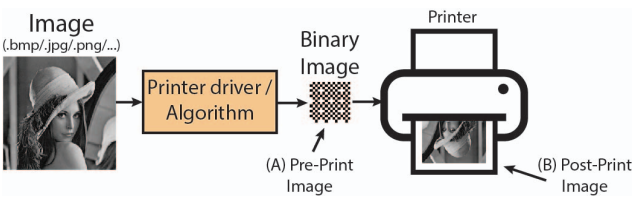


Fig. 4: Dataset of pre-print images (A) and post-print images (B).

Note that the dataset (A) has binary halftone images, and the dataset (B) has continuous-tone color halftone images. The print/scan process cannot reproduce accurately the tiny dots due to mechanical and optical limitations [1].

1) Dataset type (A) - binary halftone: To create the binary halftone dataset (shortly, D_A), we chose two image datasets: DIV2K, the reference for image up sampling [6]; and Pascal, the reference for image segmentation [4]. These datasets are composed of rectangular colored images, with different dimensions: DIV2K consists of 900 images with more than 2000 pixels of resolution (vertically or horizontally); and Pascal consists of 1872 images with a fixed dimension of 500×375.

For our tests, we chose all 900 images from the DIV2K dataset and randomly selected 900 images from the Pascal dataset, then converted all the color images to grayscale, using Python's Pillow library. After that, we crop all DIV2K images to 2000×1124 size, to regularize the shape of this dataset.

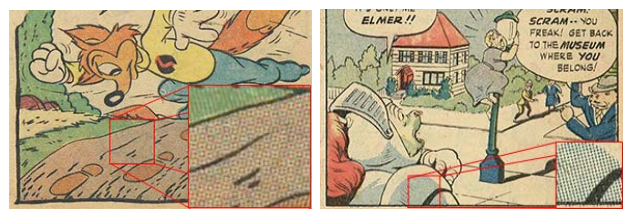
Thus, we generated two types of grayscale images G, the larger images (DIV2K) and the smaller ones (Pascal). After that, we reduced the dimensions of all images G by half using area interpolation, generating $G_{0.5}$ images. We then converted all G and $G_{0.5}$ images to binary halftone B and $B_{0.5}$ images using the Python library [10]. $B_{0.5}$ is the input image for the halftone resolution increasing algorithms and B is the ideal output.

We created four different types of halftone, as cited before: Euclidean-shaped, square-shaped, circle-shaped and triangle-shaped. To exemplify the halftoning process, we describe the Euclidean dot generation code below:

```
B = halftone(G, euclid_dot(spacing=G.shape[0]/lpi,
    angle=45))
```

Where B is the binary halftone image, and G is the original grayscale image. The *euclid_dot* function indicates the type of halftone that will be created. The parameters of the halftone function stay the same for all types of halftone. The spacing parameter varies according to the number of lines of the image, divided to the desired LPI (it can be a float number). The angle of all generated halftones was always 45 degrees.

2) Dataset type (B) - scanned halftones: To create the scanned halftones dataset (shortly, D_B), we used scanned comics from the Digital Comic Museum [12]. These comics are mainly from the mid 50's and they are printed using pattern halftones, as shown in the figure 5.



(a) Daffy Tunes Comics 012, p.7. (b) Elsie the Cow 001, p.9.

Fig. 5: Examples of D_B images, before converting them to grayscale.

We picked 900 images at random from among these comics, but only took up to two pages from the same comic, as we want to avoid creating biased dataset.

To create B and $B_{0.5}$ images in D_B dataset, we converted the scanned color images into grayscale, and for $B_{0.5}$, we resized them.

B. Training the model

We trained the machine learning algorithms using $B_{0.5}$ images as input and B as the expected output, in both datasets (D_A and D_B).

For WDZT algorithm, we kept the same parameters listed in [2].

For EDSR, we kept the same configuration pointed out in [7]: 16 residual blocks; Mean Absolute Error (MAE) loss function; and one up sampling layer, since we want up scale an image only 2×; Adam optimizer, with learning rate varying from 10^{-4} down to 5×10^{-5} by piecewise constant decay function, with the boundaries parameter set to 5000; and 100 epochs of training. The only major parameter we change on the structure of the network was the depth of input layer that originally was three to one, since the original EDSR was create to process RGB images and we are processing binary or grayscale images.

We executed the tests with 900 DIV2K, 900 Pascal and 900 comics halftone images:

- D_A - Pre-print (4 subdatasets): Euclidean-shaped, square-shaped, circle-shaped and triangle-shaped halftones;

- D_B - Post-print (1 subdataset): Grayscale halftone scanned comics.

It is a hard to do a fair comparison between EDSR and WZDT algorithms, since they have few similarities in their structure. WZDT algorithm can be trained using only a few sample images while EDSR algorithm can use much more sample images. Thus, in the first two experiments, we did only the tests that WZDT can execute, with few sample images (Table I). We followed the same training methodology of the original work [2]: in each test, we chose some random training and test images among the 900 images, in both DIV2K and Pascal and repeated this process 450 times in experiment 1 and 90 times in experiment 2.

We executed the third experiment (Table I) only for EDSR algorithm. We split the dataset into 70% images for training, 20% images for validation and 10% images for test. We repeated this training and test 5 times.

TABLE I: How much images we used in each experiment.

Experiment	WZDT (Train/Test/Repeat)	EDSR (Train/Val./Test/Repeat)
Experiment 1	9/1/450	9/2/1/450
Experiment 2	32/5/90	32/9/5/90
Experiment 3	-	630/180/90/5

For D_B dataset, we did only the experiment 3 using EDSR algorithm, because WZDT only produces binary outputs.

IV. Results

The results are organized into two categories:

- Pre-print experiments with D_A dataset. In experiments 1 and 2, we compare EDSR and WZDT algorithms. In experiment 3, we test only EDSR algorithm;
- Pos-print experiments with D_B dataset. We execute only the experiment 3 with EDSR algorithm. Furthermore, we added two more experiment: a) a comparison of our EDSR model with EDSR models trained using grayscale continuous-tone images; b) a visual comparison with other four classic up scaling algorithms.

We also show box plots to compare the algorithms. In these graphs, the lower the interquartile box is located, the greater the accuracy; and the narrower the box, the smaller the dispersion. The dashed line is the population mean value. Each point is a MAE between an ideal image and an image generated by the algorithm. Due to space limitations, we only show the images from experiments 2 and 3 for Euclidean halftones.

A. Pre-print experiments

The table II shows the average of the $n = 450$ results for each experiment in the dataset D_A. Due space limitation, we only show the results on DIV2K in figure 6. The results of experiments 1 and 2 show that the EDSR has lower MAE and standard deviation than the WZDT, indicating more stability to obtain better results in the D_A dataset.

Another information that can be seen in figures 6(a) and (b) is, the larger the number of training samples of the EDSR, the smaller the variation of the results: for EDSR, the box plots of (b) are narrower than (a). In contrast, the WZDT algorithm has the opposite behavior: despite the larger amount of training samples results in better results, the standard deviations increase, indicating an overfitting. In other words, the EDSR algorithm has better accuracy and convergence for up scaling the halftones than the WZDT. Figure 7 depicts some images generated in this experiment.

TABLE II: Results of experiments in D_A datasets. The listed values are the MAE between the processed image and the ideal image, followed by the standard deviation (MAE ± SD) of $n = 450$ experimental results.

Experiment	Dataset	Pattern	WZDT	EDSR
Experiment 1	DIV2K	Circle	0.065 ± 0.0193	0.049 ± 0.0148
		Euclid	0.060 ± 0.0196	0.046 ± 0.0119
		Square	0.069 ± 0.0201	0.041 ± 0.0126
		Triangle	0.082 ± 0.0231	0.049 ± 0.0152
	Pascal	Circle	0.064 ± 0.0212	0.049 ± 0.0170
		Euclid	0.061 ± 0.0218	0.047 ± 0.0144
		Square	0.069 ± 0.0229	0.041 ± 0.0144
		Triangle	0.083 ± 0.0254	0.049 ± 0.0172
Experiment 2	DIV2K	Circle	0.052 ± 0.0238	0.042 ± 0.0144
		Euclid	0.051 ± 0.0209	0.038 ± 0.0115
		Square	0.058 ± 0.0231	0.041 ± 0.0126
		Triangle	0.068 ± 0.0274	0.041 ± 0.0131
	Pascal	Circle	0.052 ± 0.0254	0.042 ± 0.0163
		Euclid	0.051 ± 0.0225	0.039 ± 0.0132
		Square	0.058 ± 0.0252	0.041 ± 0.0144
		Triangle	0.068 ± 0.0298	0.040 ± 0.0157
Experiment 3	DIV2K	Circle	-	0.034 ± 0.0140
		Euclid	-	0.029 ± 0.0144
		Square	-	0.034 ± 0.0136
		Triangle	-	0.035 ± 0.0139
	Pascal	Circle	-	0.034 ± 0.0155
		Euclid	-	0.029 ± 0.0158
		Square	-	0.033 ± 0.0153
		Triangle	-	0.035 ± 0.0154

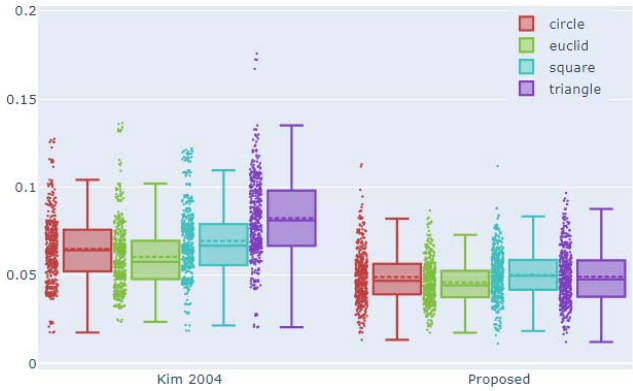
B. Post-print experiments

Table III and figure 8 show the results of the experiment 3 using D_B dataset. We executed 5 times the training, validation and test described in Table I generating $n = 90 \times 5 = 450$ results. After that, we up sampled $B_{0.5}$ images with four classic resampling algorithms: nearest neighbor, bilinear, bicubic, and Lanczos. EDSR produced the lowest error with the smallest standard deviation. Figure 10 shows that EDSR algorithm generates visually better images than the classic resampling methods.

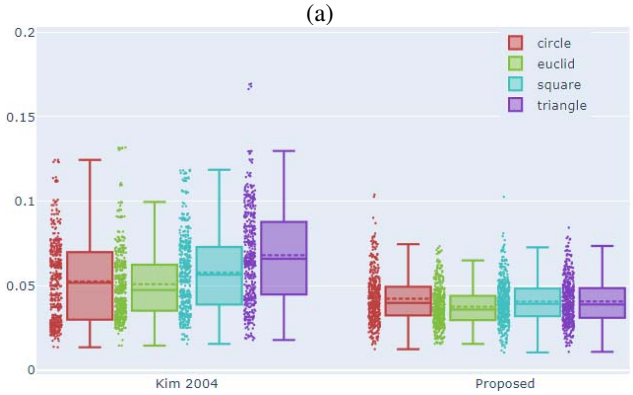
TABLE III: Results of $n = 450$ experiments using D_B post-print dataset.

Algorithm	MAE ± SD
Bicubic	0.051 ± 0.0190
Bilinear	0.056 ± 0.0193
EDSR	**0.036 ± 0.0141**
Lanczos	0.051 ± 0.0194
Nearest Neighbor	0.056 ± 0.0199

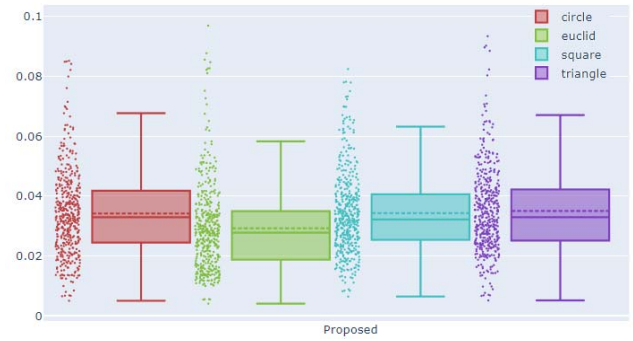
Furthermore, we trained the EDSR model with grayscale images from both Pascal and DIV2K datasets, following the proportions of experiment 3 (630 images for training, 180 for validation and repeating 5 times). Then, we tested the resulting



(a)



(b)

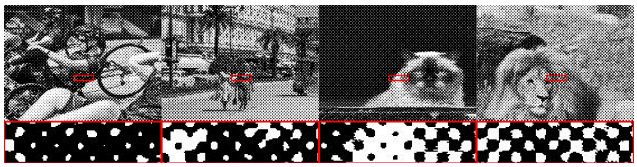


(c)

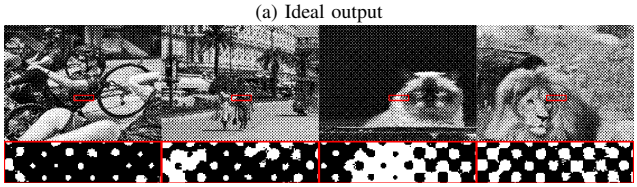
Fig. 6: Box plots of the experiments on D_A dataset with DIV2K: a) Training with 9 images and testing with 1 image (repeated 450 times); b) Training with 32 and testing with 5 images (repeated 90 times); c) Training with 630 images and testing with 90 images (repeated 5 times)

models to upsample 90 comics images from D_B 5 times with $n = 90 \times 5 = 450$ total experiments. The results depicted in table IV and figure 9 show that the model trained on comics halftone training images has better performance than the other models trained on grayscale images, indicating that "generic" grayscale upsampling models are not suitable for upsampling halftone images.

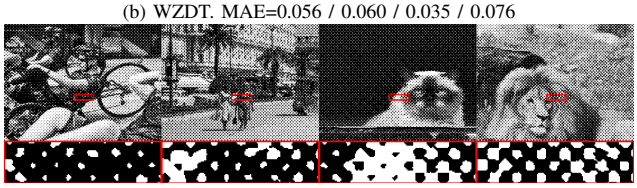
We also tested how EDSR would behave if the input of the algorithm is the ideal image from D_B, that is, the raw scanned image. For this test, we used the weights obtained



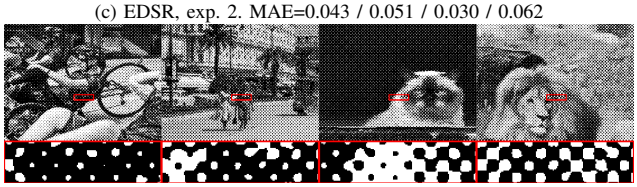
(a) Ideal output



(b) WZDT. MAE=0.056 / 0.060 / 0.035 / 0.076



(c) EDSR, exp. 2. MAE=0.043 / 0.051 / 0.030 / 0.062



(d) EDSR, exp. 3. MAE=0.039 / 0.041 / 0.024 / 0.056

Fig. 7: Outputs of DIV2K D_A euclidean dataset: a) Ideal output image; b) WZDT (experiment 2); c) EDSR (experiment 2). d) EDSR (experiment 3). MAE values are the differences between the ideal images in (a) and the processed images.

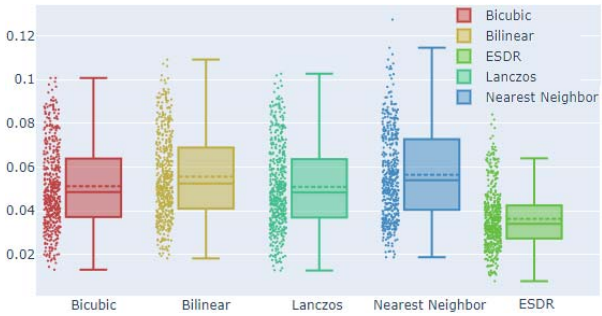


Fig. 8: Box plot comparison of the EDSR model with classic interpolations, with n = 450, tested on D_B dataset.

TABLE IV: Results obtained upsampling comics halftone images (D_B dataset) using models trained on halftone and grayscale images.

Training dataset	MAE ± SD
Comics halftone images	**0.036 ± 0.0141**
DIV2K grayscale images	0.039 ± 0.0150
Pascal grayscale images	0.043 ± 0.0151

in the experiment 3 using D_B dataset. Since we do not have the ideal up scaled image, we compare visually the results of EDSR model with the two best interpolations shown in table III: bicubic and Lanczos. The results are shown in figure 10.

The EDSR model generated images with considerably greater sharpness than the other interpolations.

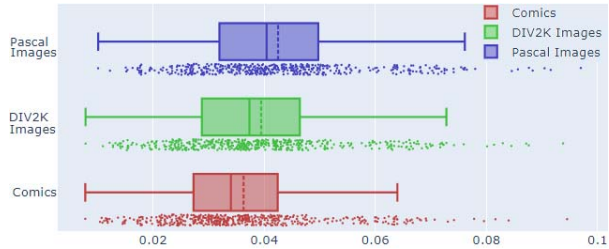
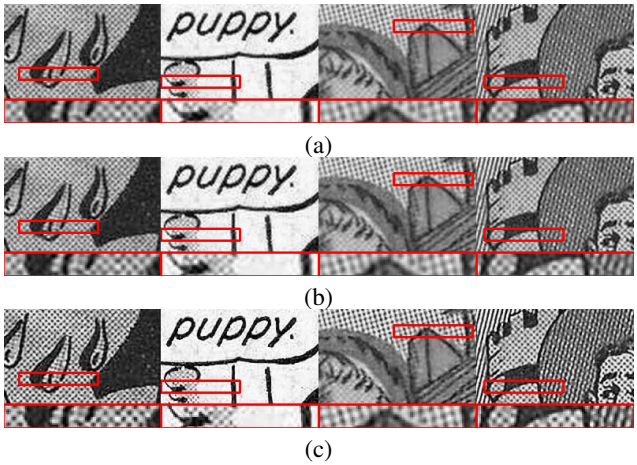


Fig. 9: Box plot comparison of the EDSR model trained on Comics halftone images versus models trained on different grayscale datasets, with $n = 450$ tests performed on the D_B dataset.



(a)

(b)

(c)

Fig. 10: Comparisons of different types of interpolations: a) bicubic; b) Lanczos; c) proposed EDSR.

V. Conclusions

This work introduced a new way to upscale resolution of halftone images. The EDSR technique yielded better results than WZDT for four types of pre-print halftones. We showed how to train properly the EDSR model to achieve the best results on post-print halftone images, that is, halftone images that has been printed and scanned. We showed that EDSR is better than the four classical resampling methods: nearest neighbor, bilinear, bicubic and Lanczos. Furthermore, the results obtained in our experiments demonstrate that scanned grayscale halftone images cannot be properly upscaled using models designed to increase the resolution of grayscale images. For halftone resolution upsampling, EDSR trained with halftone comics dataset is a better option than simple interpolations or EDSR trained with continuous tone images.

References

[1] Thrasyvoulos N Pappas. "Digital Halftoning Techniques for Printing". In: *Icps*. Vol. 94. 1994, 47th.

[2] H.Y. Kim. "Binary Halftone Image Resolution Increasing by Decision Tree Learning". In: *IEEE Transactions on Image Processing* 13.8 (Aug. 2004), pp. 1136–1146.

[3] Jerry Waite, Cheryl Willis, and Garth Oliver. "Setting halftone LPI: Taming the beasts of resolution". In: *The Technology Teacher* (Oct. 2006), pp. 1–10.

[4] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. 2012.

[5] Chao Dong et al. *Image Super-Resolution Using Deep Convolutional Networks*. 2015.

[6] Eirikur Agustsson and Radu Timofte. "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* July 2017.

[7] Bee Lim et al. *Enhanced Deep Residual Networks for Single Image Super-Resolution*. 2017.

[8] Daniel L. Lau and Gonzalo R. Arce. *Modern Digital Halftoning*. Ed. by Daniel L. Lau and Gonzalo R. Arce. CRC Press, Oct. 2018.

[9] Muhan Li and Yijian Jin. "An Hybrid Parallel Network Structure for Image Classification". In: *Journal of Physics: Conference Series* 1624.5 (Oct. 2020), p. 052005.

[10] Bohumír Zámečník. *Bzamecnik/halftone: Halftoning in python*. Nov. 2020. URL: https://github.com/bzamecnik/halftone.

[11] Tal Frank et al. "A Study on Halftoning Improvement for Low-Resolution Digital Print Engines With Machine Learning Methods". In: *IEEE Access* 10 (2022), pp. 19780–19795.

[12] Digital Comic Museum. *Digital Comic Museum*. 2022. URL: https://cdn.digitalcomicmuseum.com.