AUDITORY GRAPHS FROM DENOISING REAL IMAGES USING FULLY SYMMETRIC CONVOLUTIONAL NEURAL NETWORKS

Rodrigo F. Cádiz

Music Institute, Faculty of Arts
Department of Electrical Engineering
School of Engineering
Pontificia Universidad Catolica de Chile
Santiago, Chile
rcadiz@uc.cl

Lothar Droppelmann

Department of Computer Science
School of Engineering
Pontificia Universidad Catolica de Chile
Santiago, Chile
ldroppelmann@uc.cl

Max Guzmán

Department of Electrical Engineering
School of Engineering
Pontificia Universidad Catolica de Chile
Santiago, Chile
maguzman5@uc.cl

Cristian Tejos

Department of Electrical Engineering
School of Engineering
Pontificia Universidad Catolica de Chile
Santiago, Chile
ctejos@uc.cl

ABSTRACT

Auditory graphs are a very useful way to deliver numerical information to visually impaired users. Several tools have been proposed for chart data sonification, including audible spreadsheets, custom interfaces, interactive tools and automatic models. In the case of the latter, most of these models are aimed towards the extraction of contextual information and not many solutions have been proposed for the generation of an auditory graph directly from the pixels of an image by the automatic extraction of the underlying data. These kind of tools can dramatically augment the availability and usability of auditory graphs for the visually impaired community. We propose a deep learning-based approach for the generation of an automatic sonification of an image containing a bar or a line chart using only pixel information. In particular, we took a denoising approach to this problem, based on a fully symmetric convolutional neural network architecture. Our results show that this approach works as a basis for the automatic sonification of charts directly from the information contained in the pixels of an image.

1. INTRODUCTION

One of the most common ways for presenting numerical information is through graphs and charts. Since graphs and charts are typically contained inside digital or printed images, these figures are commonly the only publicly available representation of the original data [1]. In addition, main internet search engines nowadays include figures in their results and scientific charts are abundant not only on the web, but also embedded in all kind of digital documents [2]. A sighted person can easily understand the characteristics of the numerical information contained in a chart image inside a web page or document, but for visual impaired people, this is a significant problem, as they have no means for accessing the underlying data.

The sonification of chart data, in the form of auditory graphs, has been shown to be of great aid for visually impaired people in the understanding of numerical data [3] [4] [5]. In this paper, we focus our work in extracting the relative values of bars or the overall shape of a curve, which would allow a visually impaired user to effectively estimate the relation between bar elements or the temporal evolution of a curve in a chart.

This article is structured as follows. In section 2 we review previous work devoted to chart data extraction from images. In section 3 we discuss the most important details of our denoising-based approach. In section 4 we present the network architecture, datasets, training procedures, post-processing stages and sonification schemes. In section 5 we present the results of our approach and finally, in section 6 we reflect on our main findings and provide future lines of development for our approach.

2. PREVIOUS WORK

There have been several attempts to design systems that can automatically infer chart types from an image or extract contextual information. Traditional approaches are based on machine learning. In [6], a new heterogeneous feature extractor, denominated the heterogeneity index, is able to accurately classify charts, based on the detection of micro-structural features linked to the similarity of the chroma effects present in the images. In [7] a web miner that searches for SVG-based images on the web, based on

a random forest classifier, automatically classifies them by type (bars, lines, pies). In [8], the authors used a support vector machine algorithm to classify into ten different chart categories. In [9], shape descriptors were used in combination with a multiple-instance learning approach to classify charts. In [10], the authors analyzed the color distribution of the chart images in conjunction with a pattern matching approach based on edges for the classification into five chart types.

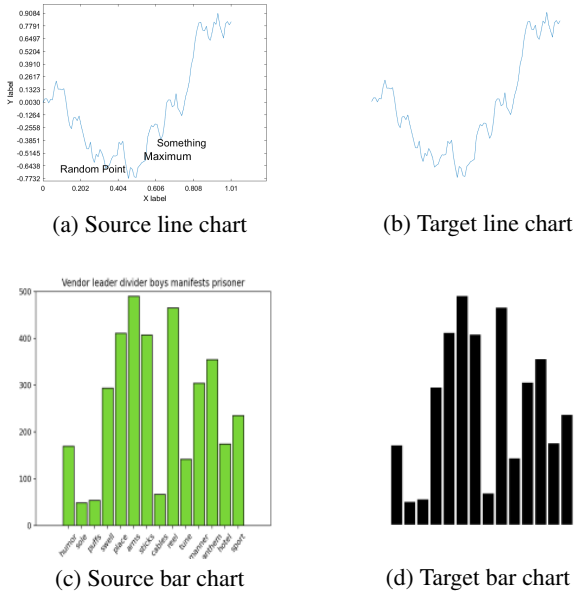
In recent times, there is increasing evidence in favor of using deep networks for this task as an alternative to traditional machine learning. In [1], the authors investigated how to automatically recover information from chart images using both a text analysis pipeline that identifies text elements present in the image and classifies their role in the chart, and a convolutional neural network that classifies the chart type (bars, points, lines, or areas). In [11], the authors used an improved version of LeNet convolutional neural network to classify images into eleven different types. In [12], a combination of convolutional neural networks and deep belief networks, allowed to achieve better scalability and stability according to the authors. In [13], the authors evaluated popular convolutional neural network architectures for automatic chart classification and found out that they outperformed methods based on conventional approaches such as support vector machines, random forest or nearest neighbor approaches. In [14], a deep learning based chart classifier achieved an accuracy of over 99%.

Previous works suggest that the task of automatic classification or extraction of contextual information from images of charts has been already addressed. However, there are only a few works specifically devoted to extracting the data itself [15]. We found a few works closely related to our approach. Scatteract [16] is a system based on a deep learning object detection model, which can automatically extract data from scatter plots, with a focus on plots with linear scales. This model is used to detect points, tick marks and tick values. It can also find the affine transformation between pixel coordinates and chart coordinates. In [17], the authors developed a system that can extract and recognize data fields in bar charts and infer the numerical data they contain. This is achieved by the multiplication of each bar's height in pixels by the y-scale-to-pixel ratio. This approach does not work for logarithmic values. In [14], the authors used the Otsu thresholding algorithm to detect the height of bars in bar charts. In [18], a region-based convolutional neural network is able to extract numerical data from pie charts. Even though these works are similar in spirit to ours, there are significant differences: our method is aimed towards real-world internet images, it does not rely on specialized or custom-designed descriptors, it is scalable, and it is based on denoising, as we will detail now.

3. A DENOISING APPROACH TO AUDITORY GRAPHS

We adopted a denoising approach for the automatic generation of auditory graphs, to facilitate the direct sonification of images containing charts. In this article, we will focus on single line and bar charts, already classified as such. Let's suppose we have access to very clean binary images, such as the ones displayed in the right column of figure 1. Then the problem of estimating a data stream from those images is a relatively straight forward computer vision problem. Of course, this approach is not suitable for lines containing multiple lines or stacked bars.

For example, in the case of the line graph of figure 1 (b) we could simply navigate through each column of the image and estimate the position of the upper non white pixel for each column in order to obtain a data series. For the chart of figure 1 (d) we can do a similar procedure and infer each bar's width from the number of non white pixels across the bottom row of the image.



(a) Source line chart (b) Target line chart

(c) Source bar chart (d) Target bar chart

Figure 1: The objective of our denoising process is to take bar or line images, e.g. (a) and (c) and transform them into clean images (b) and (d), respectively. This is achieved by considering as noise all elements that are not part of bars or lines, such as grid lines, labels and tick marks.

The problem is much more difficult when working with real-world images, e.g. figure 1 (a) and (c). As seen, both images contain a significant amount of elements that can hinder a simple approach as that described above. Real-world chart images contain bounding boxes, tick marks, text, titles, legends and labels that can occlude the pixels that carry the actual data contained in the chart. In our approach, all these elements are considered to be noise and our goal is to get rid of that noise, by a denoising process.

Our denoising approach consist of four steps (Figure 2). We consider a pre-classified image containing a line or bar chart as the input to our system. As we previously stated, we do not include a classification stage in our work. The first step is to scaled the image down to a 256 x 256 size. As we are concerned with the shape of lines and height and width of bars, all color information must be removed at this stage. Second, we apply a fully symmetric convolutional neural network, specifically designed for denoising [19], that blurs out all noisy elements that are not necessary. Once we have attenuated all the noise, we apply a binary threshold. The binary image, dependent on the particular threshold that is applied, can generate isolated pixels or disconnected elements that can be removed or joined by means of morphological filters (erosion, dilation and closing) [20]. At the end of the denoising process, we obtain a binary image that is much easier to convert into a data stream ready to be used for further sonification, as described in section 4.5.

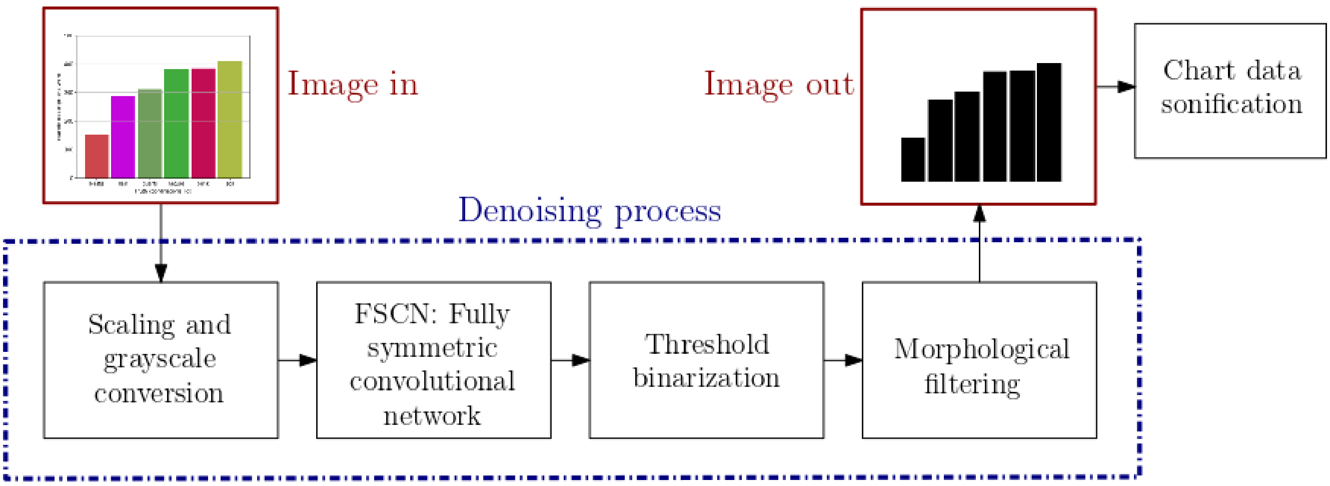


Figure 2: Our approach consists in a denoising process where all elements that are not a part of a bar or a line are removed. First, an input image is scaled down to a 256 x 256 size and all color information is removed. Second, we apply a fully symmetric convolutional neural network that blurs out all elements that are not necessary. Third, a binary threshold is applied. Fourth, all isolated pixels are removed and disconnected elements are merged by morphological filters. Finally, we obtain a binary image that is much easier to convert into a data stream ready to be used for further sonification.

4. METHODS

4.1. Network architecture

For the denoising task, we decided to implement a variation of the network architecture proposed in [19]. The architecture of the network (Table 1) consists of eight pairs of symmetric convolutional and deconvolutional layers, each of the same size (256 x 256) as the original image, with 64 channels each. Conv2D refers to a convolutional layer while Conv2DTr denotes a transposed convolutional layer, also known as a deconvolution. The left column displays layers, the middle one the shape of each layer and the right column the number of parameters of each layer. The size of the filter for each layer was 3 x 3 and all of them contained a ReLU activation function.

This architecture allows learning convolutional-deconvolutional mappings from the original chart images to the binary segmented ones without reference to image priors, and with better results than state-of-the-art denoising algorithms. According to [19]: "the convolutional layers act as feature extractor to encode primary components of the image contents while eliminating corruptions, and the deconvolutional layers then decode the image abstractions to recover the image content details". In our case, corruptions refer to unwanted items such as grids lines, tick marks, text and legends.

4.2. Training

For training, we used an Adam optimizer with a mean-squared error loss and trained in batches of 16 images. We chose this optimizer as it is very computationally efficient and works well with large datasets with little memory requirements [21]. We found the minimum loss by early stopping with a patience factor of 10, meaning that if the loss did not improve after 10 epochs, the training halts and we keep the loss and state of the network from 10 epochs in the past. We used Google Colaboratory [1] to design and train our models.

Layer	Shape	Parameters
INPUT	(256, 256, 1)	0
Conv2D	(256, 256, 64)	640
Conv2DTr	(256, 256, 64)	36928
Conv2D	(256, 256, 64)	36928
Conv2DTr	(256, 256, 64)	36928
Conv2D	(256, 256, 64)	36928
Conv2DTr	(256, 256, 64)	36928
Conv2D	(256, 256, 64)	36928
Conv2DTr	(256, 256, 64)	36928
Conv2D	(256, 256, 64)	36928
Conv2DTr	(256, 256, 64)	36928
Conv2D	(256, 256, 64)	36928
Conv2DTr	(256, 256, 64)	36928
Conv2D	(256, 256, 64)	36928
Conv2DTr	(256, 256, 64)	36928
Conv2D	(256, 256, 64)	36928
Conv2DTr	(256, 256, 64)	36928
OUTPUT	(256, 256, 1)	577

Table 1: The architecture of the neural network consists of 16 internal symmetric convolutional layers of size 256 x 256 with 64 channels each. Conv2D refers to a convolutional layer while Conv2DTr labels a transposed convolutional layer, also known as a deconvolution. The left column displays layers, the middle one the shape of each layer and the right column the number of parameters of each layer. Filter size was 3 x 3 for all layers.

[1] http://colab.research.google.com

4.3. Dataset

A complete dataset for deep learning training must include a training set, used for training the network, and a test set, consisting on images never seen by the network before, used to assess the real effectiveness of the model for new data.

In the case of bar charts, our training dataset consisted of 5000 random charts generated in Python. We varied bar widths and styles, grid sizes and scales, random text positioned in random places, tick marks and legends. For testing, we used 1000 Python and 544 real-world images downloaded from the internet.

In the case of line charts, our training dataset consisted of 4000 charts generated in Python, 4000 charts created in Matlab and 240 real-world images obtained from the internet. Both Matlab and Python charts simulate a Brownian motion in order to create a great diversity of curves. We also varied line width and styles, grid sizes and scales, random text positioned in random places, tick marks and legends. The real-world images downloaded from the internet were 40 and we used data augmentation (rotation and reflections) to obtain 240 real-world images. Our test dataset consisted of 1000 Matlab charts, 1000 Python charts and 60 real-world images.

4.4. Post-processing

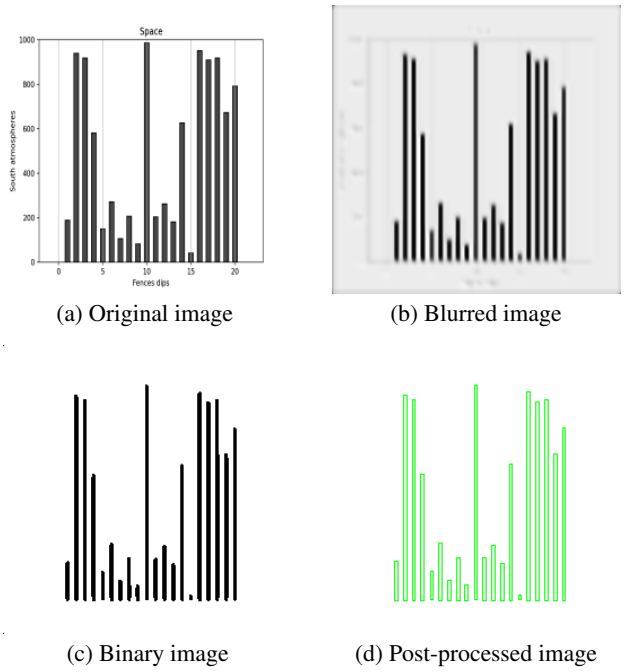
The networks we implemented are designed to blur the unwanted information as much as possible, as seen in figures 3 and 4. The original images (figures 3(a) and 4(a)) are scaled down to a 256 x 256 pixel size and converted to gray. The images are then processed by a FSCN deep model, resulting in the blurred images (figures 3(b) and 4(b)).

A threshold, tuned to the average pixel intensity, is applied to obtain binary images (figures 3(c) and 4(c)). Morphological operations, such as dilations, erosions and closings are then applied to connect isolated segments and remove pixel islands, resulting in the wanted clean images (figures 3(d) and 4(d)).

4.5. Sonification

For the sonification of the data, we proposed a simple scheme based on an iterative sweep of the pixel intensity levels of each column of the post-processed image. For each column, we can estimate the position of the upper non white pixel and we save its y-coordinate into a data array. In the case of line charts, it is probable that each column will produce a slightly different value, while in the case of bar charts, many columns will have the same y-coordinate value, as those pixels belong to the same bar. Some of the columns will only contain white pixels, signaling either bar separation of the beginning and ending of a line or curve. This simple sonification scheme, similar to the ones proposed in [22] and [23], associates pitch with the y-coordinate and time with the x-coordinate. This approach has the advantage that it works in the same fashion for both bar and line charts, allowing the user to discriminate the type of chart by simply hearing the information, without the need of extra contextual information.

This sonification scheme follows all the recommendations proposed in [5]: map the y-axis to pitch, use musical sounds instead of sine waves, consider an ambitus of midi notes within the range 35-100, and presentation at a speed that does not impair comprehension.



(a) Original image (b) Blurred image

(c) Binary image (d) Post-processed image

Figure 3: Steps of the denoising process. The original image (a) is denoised by an FSCN deep model, resulting in the image (b). Then, a 50% threshold is applied to obtain a binary image (c). Morphological filters are then applied to connect isolated segments and remove pixel islands, resulting in the post-processed image (d).

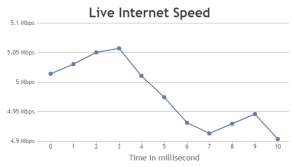
5. RESULTS

When testing the FSCN networks, we achieved an optimal loss of 0.007 after 30 epochs for the bar charts and 0.0047 for the line charts after 12 epochs. To assess the quality of our denoising system, including post-processing, we calculated the average mean squared error (MSE) and its standard deviation between segmented images directly from the originals, and the ones generated by our method (table 2). We chose this metric as a way to determine the quality of the sonification input, as we need to eliminate all objects that are not data. In consequence, this error allows us to establish a criteria of how well we are able to effectively eliminate unneeded elements from the charts.

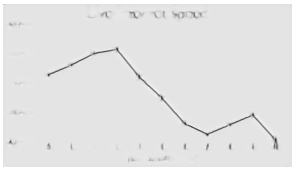
N	Dataset	Average MSE	Standard deviation MSE
999	Python	0.006	0.001
999	Matlab	0.01	0.005
60	Internet	0.03	0.02

Table 2: Mean squared error (MSE) for synthetic (Python, Matlab) and real (Internet) test datasets

For these measurements, we considered 999 synthetic chart images generated in Matlab, 999 in Python, and 60 images downloaded from the internet. These images belong to the test set, meaning that they were never seen in the training by the FSCN network. As shown in table 2, the error increases for real images, as well as the standard deviation of the error. This means that at

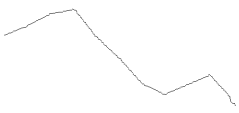


(a) Original image



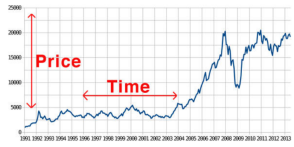
(b) Blurred image

(c) Binary image



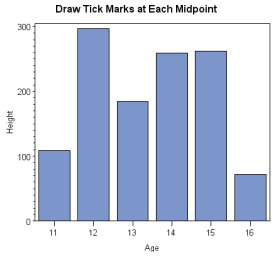
(d) Post-processed image

Figure 4: Steps of the denoising process. The original image (a) is denoised by an FSCN deep model, resulting in the image (b). Then, a 50% threshold is applied to obtain a binary image (c). Morphological filters are then applied to connect isolated segments and remove pixel islands, resulting in the post-processed image (d).

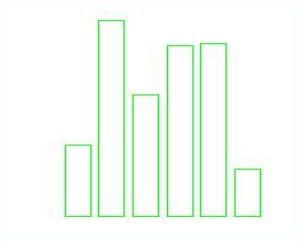


(a) Source line chart

(b) Generated line chart



(c) Source bar chart



(d) Generated bar chart

Figure 5: Examples of results. Our approach is able to generate the charts seen in the right column (b and d) directly from the source images shown in the left column (a and c). In these particular cases, these source images were obtained from the internet. All elements that are not part of bars or lines, such as grid lines, labels and tick marks, were successfully removed from the original images.

this stage our system is performing well with synthetic images, but there is still room for improvement for real-world ones. We have identified that most of the errors are generated at the borders of the images or when the images contain a significant amount of text or legends occluding the data. This is not surprising considering the fraction of real-world images that we used for training compared to the total. Our sonification stage takes into account some of these sources of errors in order to minimize their influence in the final generated sound.

However, when looking at some real-world examples, we can see that, in most of the cases, our approach performs well. Figure 5 shows two examples of our results. As it can be observed, our approach is able to generate clean and accurate bar and line charts removing all elements that are not part of bars or lines, such as grid lines, labels and tick marks.

As we explained in section 4.5, our sonification approach naturally permits the differentiation of line and bar charts, and it is significantly eased by the very clean binary images that our system produces. Reviewers can find code, repositories and demo videos showing examples of sonifications for several different kinds of charts at our github site: `https://sonificationUC.github.io`.

6. CONCLUSIONS AND FUTURE WORK

Even though these are preliminary results, based on a relatively simple FSCN deep network architecture and small datasets, we have shown that this approach could result in a very useful tool for automatic auditory graph generation directly from the pixel information of bar and line chart images. In summary, we believe that a denoising approach is a viable path for this task.

We decided to focus on a data extraction process and not on a previous classification task, as we found that there is plenty of research devoted to that topic and much fewer work is targeted to the data extraction. Additionally, we believe that our sonification approach allows the user to infer the type of chart (e.g. bar, line, etc.) directly from the generated sound.

In this paper we were not focused in the extraction of the absolute numerical quantities contained in a chart, but rather in the relative values of bars or the estimation of the overall shape of a curve. The extraction of the absolute values, which is much more complex task, is planned for future work.

We acknowledge that there is room for additional improvements. In terms of the current errors of our approach for real-world images, we have identified that most of the errors are generated at the borders of the images or when the images contain a significant amount of text or legends occluding the data. To improve performance, we are looking forward to dramatically increase the sizes and variety of our datasets, to refine the network architecture, to include robust text analysis, contextual information and color, to incorporate chart classification stages and to implement several alternative sonification strategies, with the aim of combining all of these items into a usable and freely available auditory graph tool for visually impaired users.

Finally, it is of utter importance to conduct user tests, specifically aimed towards visually-impaired people, to determine the usability and precision of our approach from a perceptual standpoint.

7. ACKNOWLEDGMENTS

This research was partially funded by Fondecyt Grants #1191710, #1161328 and Anid/PIA/Anillo ACT192064

8. REFERENCES

[1] J. Poco and J. Heer, “Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images,” in *Eurographics Conference on Visualization (EuroVis)*, vol. 36, no. 3. Blackwell Publishing Ltd, 6 2017, pp. 353–363.

[2] R. A. Al-Zaidy and C. L. Giles, "Automatic extraction of data from bar charts," *Proceedings of the 8th International Conference on Knowledge Capture, K-CAP 2015*, 2015.

[3] F. Delogu, M. Palmiero, S. Federici, C. Plaisant, H. Zhao, and O. Belardinelli, "Non-visual exploration of geographic maps: does sonification help?" *Disability & Rehabilitation: Assistive Technology*, vol. 5, no. 3, pp. 164–174, 2010.

[4] M. A. Alonso-Arevalo, S. Shelley, D. Hermes, J. Hollowood, M. Pettitt, S. Sharples, and A. Kohlrausch, "Curve shape and curvature perception through interactive sonification," *TAP*, vol. 9, no. 4, pp. 1–19, 2012.

[5] L. Brown, S. Brewster, S. Ramloll, and R. Burton, "Design guidelines for audio presentation of graphs and tables," 2003. [Online]. Available: http://sci-hub.cc/http://eprints.gla.ac.uk/3196/

[6] P. Mishra, S. Kumar, and M. K. Chaube, "ChartFuse: a novel fusion method for chart classification using heterogeneous microstructures," *Multimedia Tools and Applications*, pp. 10 417–10 439, 2020.

[7] L. Battle, P. Duan, Z. Miranda, D. Mukusheva, R. Chang, and M. Stonebraker, "Beagle: Automated extraction and interpretation of visualizations from the web," in *Conference on Human Factors in Computing Systems - Proceedings*, vol. 2018-April. Association for Computing Machinery, 4 2018.

[8] M. Savva, N. Kong, A. Chhajta, F. F. Li, M. Agrawala, and J. Heer, "ReVision: Automated classification, analysis and redesign of chart images," *UIST'11 - Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, pp. 393–402, 2011.

[9] W. Huang, S. Zong, and C. L. Tan, "Chart image classification using multiple-instance learning," *Proceedings - IEEE Workshop on Applications of Computer Vision, WACV 2007*, pp. 5–10, 2007.

[10] A. Mishchenko and N. Vassilieva, "Model-based chart image classification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6939 LNCS, no. PART 2, pp. 476–485, 2011.

[11] J. Amara, P. Kaur, M. Owonibi, and B. Bouaziz, "Convolutional neural network based chart image classification," *Computer Science Research Notes*, vol. 2703, no. May, pp. 83–88, 2017.

[12] B. Tang, X. Liu, J. Lei, M. Song, D. Tao, S. Sun, and F. Dong, "DeepChart: Combining deep convolutional networks and deep belief networks in chart classification," *Signal Processing*, vol. 124, pp. 156–161, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.sigpro.2015.09.027

[13] P. Chagas, R. Akiyama, A. Meiguins, C. Santos, F. Saraiva, B. Meiguins, and J. Morais, "Evaluation of Convolutional Neural Network Architectures for Chart Image Classification," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2018-July, 2018.

[14] W. Dai, M. Wang, Z. Niu, and J. Zhang, "Chart decoder: Generating textual and numeric information from chart images automatically," *Journal of Visual Languages and Computing*, vol. 48, pp. 101–109, 2018. [Online]. Available: https://doi.org/10.1016/j.jvlc.2018.08.005

[15] K. Davila, S. Setlur, D. Doermann, U. K. Bhargava, and V. Govindaraju, "Chart Mining: A Survey of Methods for Automated Chart Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8828, no. c, pp. 1–1, 2020.

[16] M. Cliche, D. Rosenberg, D. Madeka, and C. Yee, "Scatteract: Automated extraction of data from scatter plots," 4 2017. [Online]. Available: http://arxiv.org/abs/1704.06687http://dx.doi.org/10.1007/978-3-319-71249-9_9

[17] R. A. Al-Zaidy and C. L. Giles, "A Machine Learning Approach for Semantic Structuring of Scientific Charts in Scholarly Documents," *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pp. 4644–4649, 2017. [Online]. Available: http://aaai.org/ocs/index.php/IAAI/IAAI17/paper/view/14275

[18] P. De, "Automatic Data Extraction from 2D and 3D Pie Chart Images," *Proceedings of the 8th International Advance Computing Conference, IACC 2018*, pp. 20–25, 2018.

[19] S. A. Priyanka and Y. K. Wang, "Fully symmetric convolutional network for effective image denoising," *Applied Sciences (Switzerland)*, vol. 9, no. 4, 2019.

[20] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, third edit ed. Prentice Hall, 2008.

[21] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 12 2015. [Online]. Available: https://arxiv.org/abs/1412.6980v9

[22] A. Aparicio and R. F. Cádiz, "Audiograph: Auditory graphs for Google Sheets with Faust," in *The 24th International Conference on Auditory Display (ICAD 2018)*, 2018, pp. 1–4.

[23] L. Harrar and T. Stockman, "Designing auditory graphs overviews: An examination of discrete vs. continuous sound and the influence of presentation speed," in *Proceedings of the 13th International Conference on Auditory Display (ICAD2007)*, 2007, pp. 299–305. [Online]. Available: http://sci-hub.cc/https://smartech.gatech.edu/handle/1853/49990