– Supplementary Material –

Interactive Context-Preserving Color Highlighting for Multiclass Scatterplots

KECHENG LU, Shandong University, China
KHAIRI REDA, Indiana University-Purdue University Indianapolis, United States
OLIVER DEUSSEN, University of Konstanz, Germany
YUNHAI WANG*, Shandong University, China

ACM Reference Format:
Kecheng Lu, Khairi Reda, Oliver Deussen, and Yunhai Wang. 2023. Interactive Context-Preserving Color Highlighting for Multiclass Scatterplots. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3544548.3580734

This **supplementary material** provides additional experimental results for our submitted paper titled "Interactive Context-Preserving Color Highlighting for Multiclass Scatterplots".

Navigation:

*corresponding author

Manuscript submitted to ACM

Probability distribution for finding lightness value. To rapidly produce homogeneous backgrounds, we set a large probability for accepting a uniform lightness for all colors initially and decrease it as the number of iterations increases. As shown in Fig.1, at the beginning of the simulated annealing algorithm, the main process is to find the best uniform lightness for all colors. Then the probability is decreased according to the palette score and the number of iterations. Finally, we got the best lightness and mainly disturb the lightness of each color.

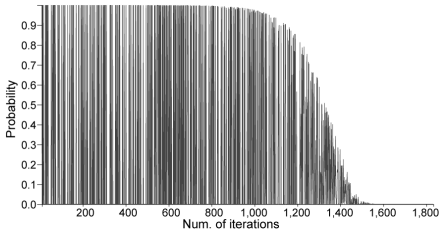

Fig. 1. The probability distribution of simulated annealing algorithm for finding the best lightness value.

Teaser with alpha blending. Due to the limited space of the paper, the full methods including *Palettailor with alpha blending* are shown in this supplementary material. One straightforward way to preserve the context during highlighting is to modulate a visual factor (e.g., opacity) of non-selected data points. Yet, this method often leads to misleading colors in overlapping regions due to alpha blending, resulting in poor class separability (see the bottom in Fig.2 (b)).

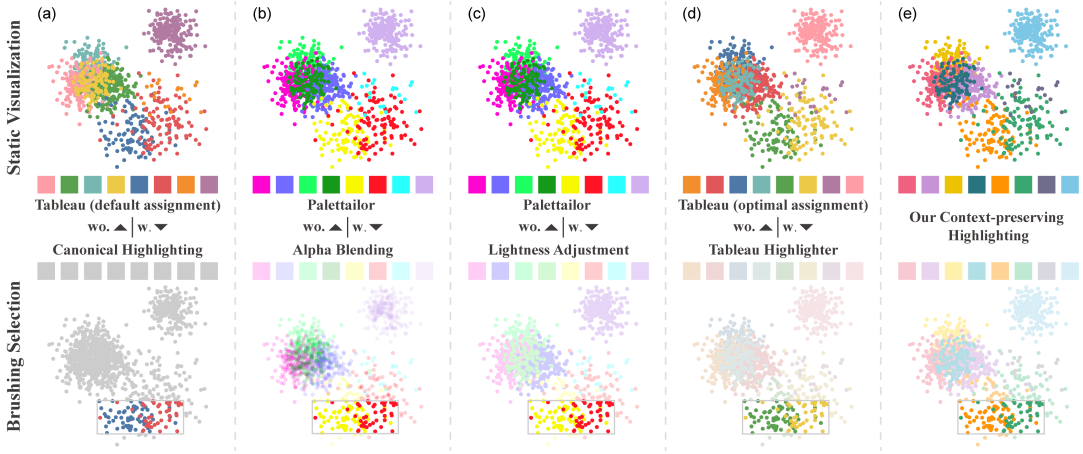

Fig. 2. Results for applying different color-based highlighting methods to brush a multi-class scatterplot. **(a)** (top) result colorized by the Tableau palette and the default assignment; (bottom) a highlighting effect achieved by assigning a grey color to all non-selected data points; **(b)** (top) result colorized by a Palettailor-generated palette [4]; (bottom) a highlighting effect is achieved by reducing the opacity of non-selected data points; **(c)** (top) result colorized by a Palettailor-generated palette [4]; (bottom) a highlighting effect is achieved by increasing the lightness of non-selected data points; **(d)** (top) result colorized by the Tableau palette and the optimal assignment; (bottom) achieving a highlighting effect by applying Tableau Highlighter function; **(e)** (top) result colorized by our method with the salient color palette; (bottom) our highlighting result by combining salient and faint color palettes. Our method allows highlighting a subset of data points while maintaining the discriminability of all non-selected points and color consistency of all pairs of color.

Detailed Discussion of the Results.

We evaluated the effectiveness of our approach against the benchmark conditions through two crowdsourced experiments for two different scenarios (static visualization and interactive exploration). For the performance of the *counting task* for static visualization, as shown in Fig.3, we found that first, *Palettailor* outperformed the two *Tableau* conditions and *Our Method (static)*. This is reasonable since the design goal of *Palettailor* is to maximize class discriminability of a scatterplot while *Tableau* is a designer-crafted palette for commonly used highlighting tasks and *Our Method (static)* also serves such tasks. *Tableau with optimized assignment* achieves better performance than *Tableau with default assignment*, this indicates that the optimal discrimination assignment approach [6] improves the discriminability of the static visualization, as shown in Figs.3 (b, c). Specifically, *Our Method (static)* seems to be slightly better than *Tableau with optimal assignment*. The results suggest that while *Palettailor* beats our method in the *counting task* for the global discriminability, the disadvantage for our method is not substantial, which represents a small overhead to pay for the ability to emphasize the desired class.

Fig. 3. One of the four six-class scatterplots used in the two experiments. There are four different colorization methods for the counting task in the top row: (a) Result generated from Palettailor; (b, c) Result generated by Tableau 10 palette with default and optimal assignment; (d) Our method result for static visualization. There are five different highlighting methods for the selecting task in the bottom row, the circles are randomly placed around the highlighted class: (e, f) using lightness adjustment and alpha blending to highlight the yellow class, the original palette is from (a); (g, h) using Tableau Highlighter to highlight the desired class based on (b) and (c), respectively; (i) Our method result for interactive visualization. The correct answer for each scatterplot is shown in the top right.

For interactive exploration, our method shows a better performance. In the *highlighting task*, we found that without informing the participants what an emphasized class is, there's a significant difference between *Our Method (interactive)* and some benchmark conditions (*Palettailor with lightness adjustment*, *Palettailor with alpha blending* and *Tableau Highlighter with default assignment*). This indicates that, with regard to the common highlighting task, our contextual highlighting method performs better than standard highlighting methods. *Palettailor with alpha blending* did not get a good highlighting performance, one reason is that colors from Palettailor might have a similar lightness to the background, as shown in Fig.3 (f). The other reason is that the mixed color from alpha blending will be attractive, such as the red and blue classes shown in Fig.10 (b).

We also found that *Our Method (interactive)* has a similar performance with *Tableau Highlighter with optimal assignment*, which implies that good discriminability helps the *highlighting task* as well. For the *matching task*, *Our Method (interactive)* performed better than *Tableau Highlighter with default assignment* and *Tableau Highlighter with optimal assignment*, while achieving similar performance to *Palettailor with lightness adjustment* and *Palettailor with alpha blending*. A possible explanation is that our method and lightness adjustment methods only perturb the lightness axis while maintaining hue and saturation. As for the alpha blending method, the major reason is that the background is white and the alpha blending does not change the color hue. An example can be found in Fig.3 (f), where the green and blue classes are similar to the original color in Fig.3 (a). However, given *Our Method* that can maintain name similarity for the de-emphasized colors, it slightly outperforms *Palettailor with lightness adjustment* and *Palettailor with alpha blending*. An example illustration can be found in Fig.3 (e, f, i). In the *selecting task*, we found that *Our Method (interactive)* achieves the best performance among all benchmark conditions, while there's no significant difference to *Palettailor with alpha blending*. The explanation of this result is similar to the *matching task*. However, *Our Method (interactive)* takes a shorter time than the alpha blending method, since the alpha blending method blends the colors of overlapping marks, potentially introducing new colors, participants might be confused about these new colors, as shown in Fig.3 (f).

Class number analysis and speed-accuracy analysis for each task of the formal study.

Counting task. To better understand how the different methods compare as the number of classes increases, we conducted a class number analysis for this class number counting task. As shown in Figs.4(a, b, c, d), we draw the confidence interval plots for the whole data and different class numbers. We can see that for different settings, the error rate and response time have similar performance. However, we found that in 10-class scatterplots, our method consumed more time than other methods while achieving less error rate, which gives an explanation for why *Palettailor* always have a low error rate and high response time: users tend to spend more time to count how many classes are there in the scatterplot when the classes have a good separability. We did not find significant interaction effects between colorization methods and cluster number ($F(3, 1432) = 0.1342; p > 0.1$). This means that the effectiveness of different methods on *counting task* seems insensitive to the configuration of the cluster number. We also provide a speed-accuracy analysis to show whether a speed-accuracy tradeoff exists. Since the relative error is categorical but not dichotomous (0/1), we conducted a Kruskal-Wallis test, where $Kruskal - Wallischi - squared = 1134.2, df = 1130, p = 0.4596$. The results indicate that the error and response time are weakly correlated. We also used some boxplots to inspect our data visually. If the data are weakly correlated, there will be a lot of overlap between the boxes. As shown in Fig.4(e), the boxes are overlapped heavily.

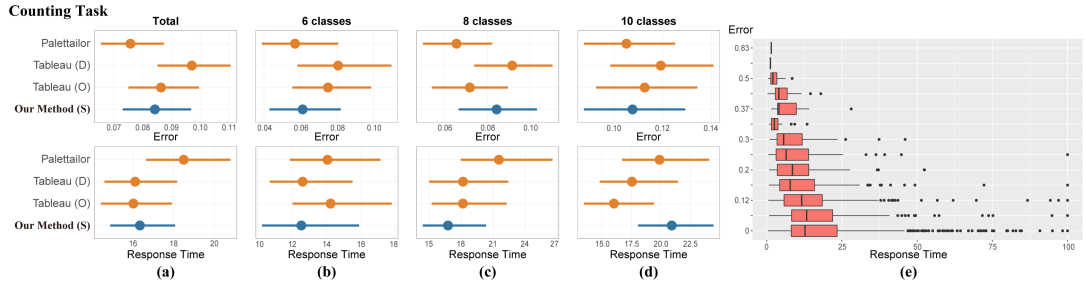

Fig. 4. Statistics for the counting task. (a) Confidence interval plots for the whole trial data of different class numbers; (b) confidence interval plots for trial data which class number six; (c) confidence interval plots for trial data which class number is eight; (d) confidence interval plots for trial data which class number is ten; (e) box plots for the whole trial data of different class number. Each table shows the statistical test results of our experimental condition (*Our Method (S)*) with the three benchmark conditions (*Palettailor*, *Tableau (D)* and *Tableau (O)*), showing the mean with 95% confidence interval ($\mu \sim$ 95%CI), W-value and p-value from the Mann-Whitney test, as well as effect size (d ~ 95%CI).

Selecting task. The analysis is similar to the global discrimination task. As shown in Figs.5(a, b, c, d), we draw the confidence interval plots for the whole data and different class numbers. We can see that for different settings, the error rate and response time have similar performance. We did not find significant interaction effects between colorization methods and cluster number ($F(4, 1790) = 0.5798; p > 0.1$). This means that the effectiveness of different methods on *selecting task* for local discrimination seems insensitive to the configuration of the cluster number. We also provide a speed-accuracy analysis to show whether a speed-accuracy tradeoff exists. Since the relative error is categorical but not dichotomous (0/1), we conducted a Kruskal-Wallis test, where $Kruskal - Wallischi - squared = 1343.4, df = 1310, p = 0.2548$. The results indicate that the error and response time are weakly correlated. We also used some boxplots to inspect our data visually. If the data are weakly correlated, there will be a lot of overlap between the boxes. As shown in Fig.5(e), the boxes are overlapped heavily.

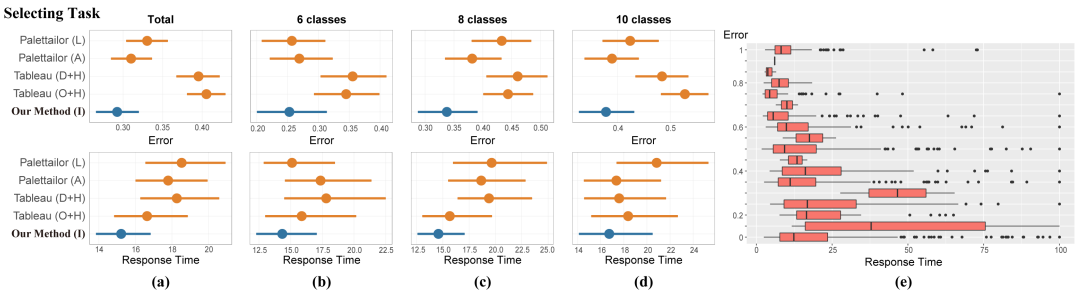

Fig. 5. Confidence interval plots and statistical tables for the selecting task. Error bars represent 95% confidence intervals. Each table shows the statistical test results of our experimental condition with the benchmark conditions (*Palettailor (L)* indicates *Palettailor with lightness adjustment*, *Palettailor (A)* indicates *Palettailor with alpha blending*, *Tableau (D+H)* indicates *Tableau Highlighter with default assignment*, *Tableau (O+H)* indicates *Tableau Highlighter with optimal assignment*, *Our Method (I)* indicates *Our Method (interactive)*).

Highlighting task. The analysis is similar to the previous tasks. As shown in Figs.6(a, b, c, d), we draw the confidence interval plots for the whole data and different class numbers. We can see that for different settings, the error rate and response time have similar performance. We did not find significant interaction effects between colorization methods and cluster number ($F(4, 1790) = 0.2685; p > 0.1$). This means that the effectiveness of different methods on *highlighting task* seems insensitive to the configuration of the cluster number. We also provide a speed-accuracy analysis to show whether a speed-accuracy tradeoff exists. Since the error of this task is dichotomous (0/1), we conducted a two-sample Wilcoxon rank sum test, where $W = 318223, p = 0.001926$. The results indicate that the error and response time are strongly correlated. We also used some boxplots to inspect our data visually. If the data are strongly correlated, there will be a small overlap between the boxes. As shown in Fig.6(e), the error(1) box has a larger response time than the correct(0) box. This is aligned with our experience: when the object is hard to find, it will take more time than an easier one.

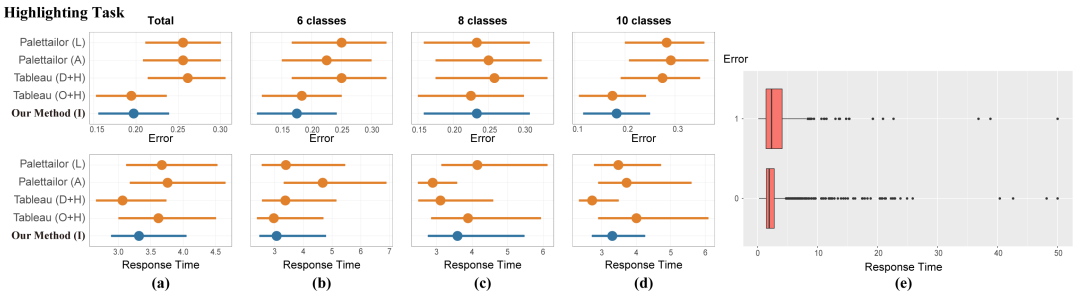

Fig. 6. Statistics for the highlighting task. Similar to the previous task.

Matching task. The analysis is similar to the highlighting task. As shown in Figs.7(a, b, c, d), we draw the confidence interval plots for the whole data and different class numbers. We can see that for different settings, the error rate and response time have similar performance. We did not find significant interaction effects between colorization methods and cluster number ($F(4, 1790) = 2.163; p > 0.05$). This means that the effectiveness of different methods on *constancy task* seems insensitive to the configuration of the cluster number. We also provide a speed-accuracy analysis to show whether a speed-accuracy tradeoff exists. Since the error of this task is dichotomous (0/1), we conducted a two-sample Wilcoxon rank sum test, where $W = 333316, p = 0.002$. The results indicate that the

error and response time are strongly correlated. We also used some boxplots to inspect our data visually. The boxes in Fig.7(e) have a large overlap, while the correct(0) box has a much larger number of outliers than the error(1) box. A possible explanation is that, as for the color constancy task, users often need to speculate the correct answer based on other colors, e.g., trials from *Palettailor with lightness adjustment*, *Palettailor with alpha blending* and *Our Method(Interactive)* have better performance on this task, while some color still needs to be speculated from the context, results in more completion time but smaller error rate, while *Tableau* methods will lead to colors that hard to be distinct and user might give up quickly, thus takes less time.

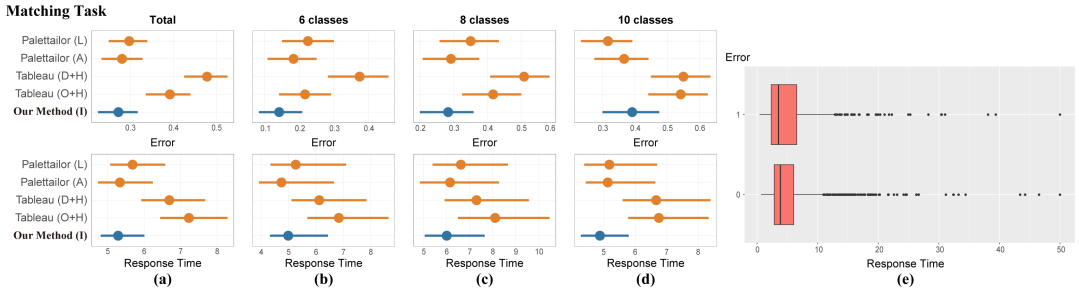

Fig. 7. Statistics for the color matching task. Similar to the previous task.

Interactive System and Extensions.

To aid designers in flexibly crafting categorical color palettes with contextual highlighting effects, we developed a web-based design tool that embodies our methodology[1]. The interface consists of four coordinated views: (i) a control panel, (ii) a palette panel for showing color information or adjusting the classes of interest, (iii) a visualization panel to provide a preview of the colorization result, and (iv) a history widget (see Fig. 8 for a screenshot). After uploading a labeled dataset, the system automatically finds an optimal color mapping scheme to colorize the input data. Classes are displayed on the palette panel, enabling the user to interactively highlight classes of interest. This adjustment forces the tool to automatically combine a new color palette from the pre-generated two palettes to emphasize the important classes. Besides, the user can directly select a subset of points from the visualization panel, using interactions like clicking or brushing, as shown in Fig. 9. The user can then save the resulting scheme using the history widget for future reference.

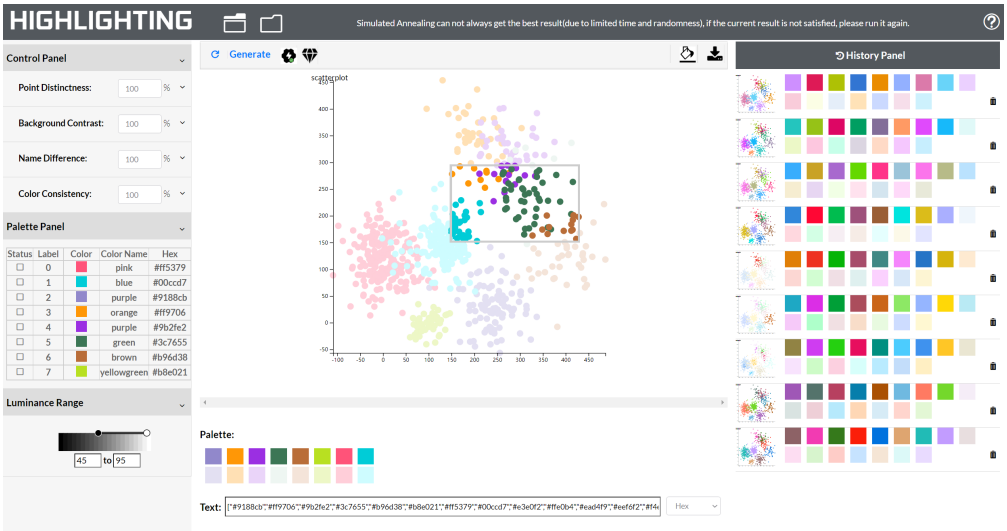

Fig. 8. Screenshot of our interactive colorization system, which consists of four panels: (i) control panel; (ii) palette panel; (iii) visualization panel; and (iv) a history panel.

[1]https://anon-link.github.io/highlighting/

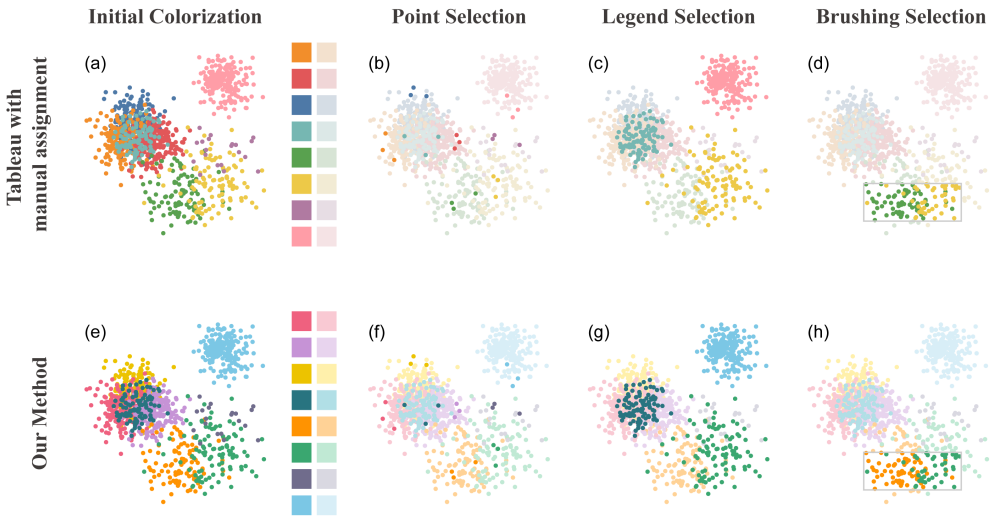

Fig. 9. Different selections for the initial colorizations (a, e), including (b, f) point selection; (c, g) legend selection; (d, h) brushing selection. Our method achieves better or at least similar performance on different tasks than Tableau.

Case Study for Single Scatterplot. Here, we analyzed the MNIST database of handwritten digits provided by Yann et al. [3], which contains 784 data dimensions with ten classes. We project this dataset using tSNE onto a 2D scatterplot with 1000 random distinct samples. We first colorized the visualization using Tableau [5] (see Fig. 10(a)-top) with the default settings. To explore the details of the data distribution, the user often uses brushing to select interesting regions. Fig. 10(a)-bottom shows the brushing result from D3 or Vega-Lite, which simply applies a grey color to de-emphasized regions. There're two problems with this strategy: first, we lose all neighborhood information; second, the original grey color from the palette is similar to the de-emphasized grey color. The discriminability of the default assignment is not enough as well. Then we tried the state-of-the-art automated colorization algorithm Palettailor [4] (see Fig. 10(b)-top), and applied alpha blending to de-emphasize the surrounding regions, as shown in Fig. 10(b)-bottom, where the distribution of the light pink and yellow classes are hard to figure out and the light blue and cyan classes are mixed together. Thanks to the optimal discrimination assignment approach [6], we can improve the discriminability of the Tableau palette (see Fig. 10(c)-top), then we can use Tableau's Highlighter to interactively emphasize desired regions (see Fig. 10(c)-bottom). While the emphasis effect is good, the details of the other classes are difficult to distinguish, e.g., the pink class and red class are hard to distinguish. In contrast to Tableau and Palettailor, our method is able to produce consistently good pop-out effects, with the emphasized region varied interactively as desired (see Fig. 10(d)). Additionally, class separability in our method is overall better than Tableau's Highlighter, e.g., the separability of pink and red classes in Fig. 10(d)-bottom is better than the pink and red classes in Fig. 10(c)-bottom. Uniquely, our method maintains good color consistency regardless of which class is being highlighted.

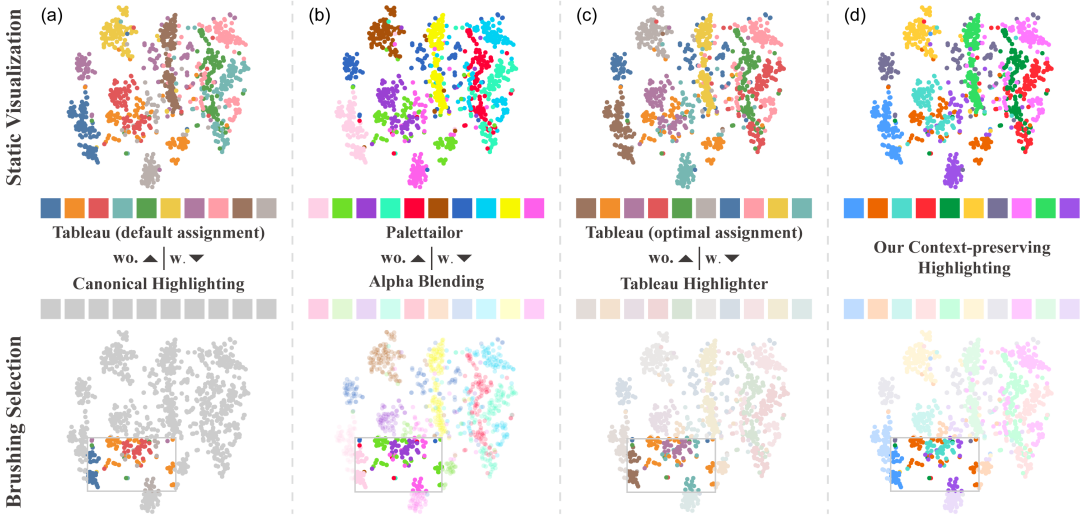

Fig. 10. Visualizing the MNIST dataset [3] with different methods for static visualizations (top row) and brushing selection (bottom row). **(a)** (top) Applying Tableau with default assignment to the data; (bottom) a highlighting effect achieved by assigning a grey color to all non-selected data points; **(b)** (top) using Palettailor to colorize the data; (bottom) applying alpha blending to highlight selected points; **(c)** (top) applying Tableau with optimal assignment; (bottom) using Tableau Highlighter to pop out the selected region; **(d)** (top) our method result for static visualization; (bottom) the corresponding highlighting result. Our method (d) gives a good highlighting effect while maintaining class discriminability during interactive exploration.

Case Study for Scatterplot Matrix. We conducted a second case study with a real-world dataset, this time using a scatterplot matrix. Here, we analyzed a subset of FORCE 2020 Well well log and lithofacies dataset for Machine Learning competition [1], which is used to predict lithology from existing labeled data using well log measurements. To simplify the visualization, we only show a handful of variables from the dataset, including *RHOB*, *GR*, *NPHI*, *DTC*, and *LITH*. The first four of these five variables are numeric, and the last is categorical, which will be used as the class information.

Figs. 11(a, b, c) shows the scatterplot matrix colorized using Our Method(Static), Tableau with random assignment, and Tableau with optimal assignment, respectively. We can see that the optimal assignment has better discriminability, but the classes in the corner are hard to be distinct, such as the green and light blue classes. Our method for static visualization achieves the best discriminability among these three results. Figs. 11(d, e, f) shows the brushing selection results from the interactive exploration, the selected area is indicated by a black rectangle. We can see that Vega-Lite achieves the best highlighting but loses all the neighborhood information, while the emphasis effect of Tableau's Highlighter is good but the details of other classes are difficult to distinguish. By comparison, the class separability of our method is better than Tableau for a baseline visualization (see Fig. 11(d). With a focus on the interesting area, our method achieves better overall class discriminability than Tableau (Fig. 10(f)), allowing the user to still investigate any de-emphasized classes. The ability to interactively vary the highlight while still maintaining context makes our method especially suited for visual exploration.

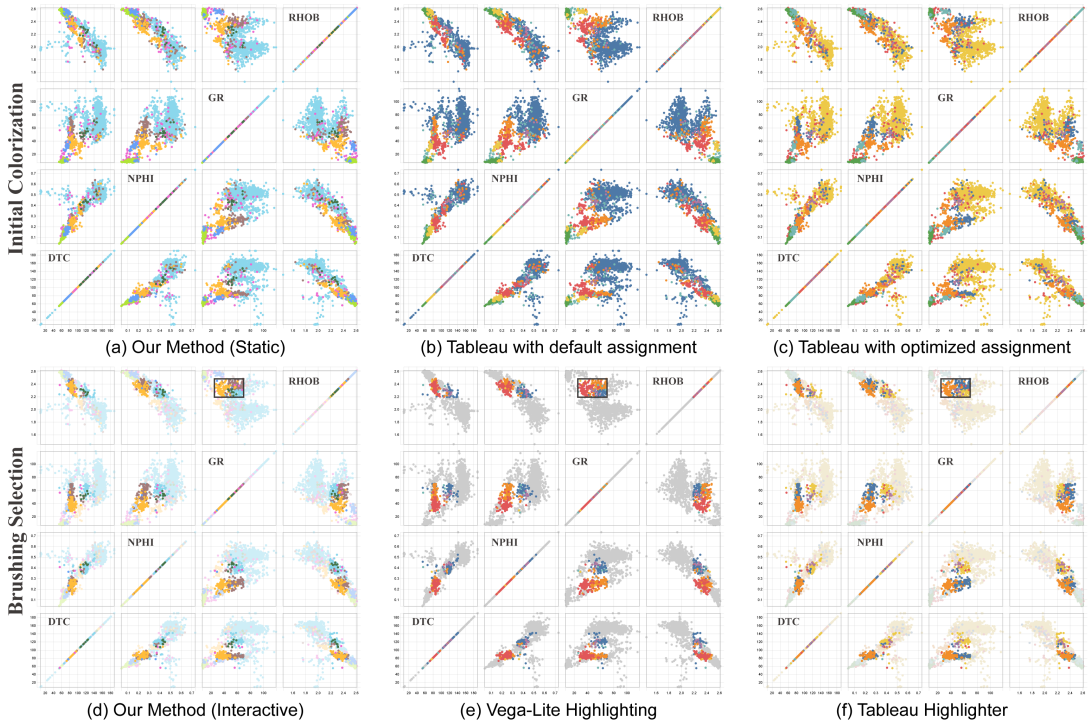

Fig. 11. Visualizing the FORCE 2020 Well dataset [1] with different methods for static visualizations (top row) and brushing selection (bottom row).

Extensions for Bar and Line Charts.. In addition to scatterplots, our color mapping method works also for other categorical visualization types such as bar or line charts. This is achieved by treating each bar or line segment as a point and then using the same method to compute their class contrasts. This allows marks of interest to be highlighted while maintaining discriminability among all classes.

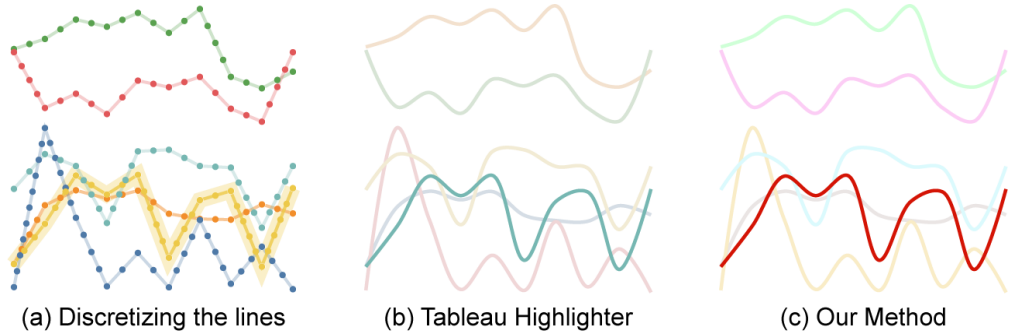

Fig. 12. Extension for line charts in (a) with the yellow line to be highlighted. (a) Discretizing each line yields a point-based representation, which we take as input for our method; (b) result generated by using Tableau Highlighter; (c) result generated by our automated contextual highlighting method. Notice that higher discriminability between the non-highlighted classes in our method.

Fig. 12 shows an example of colorizing a line chart. The highlighting effect of our automatic generation method in Fig. 12(c) is not worse than an existing designer-crafted palette from Tableau-10 with Highlighter (see Fig. 12(b)). Also here the discriminability between the other lines in the chart is maintained.

Case Study for Line Chart. We conducted a second case study with a real-world dataset, this time using line charts. Here, we analyzed an air quality dataset provided by Vito et al. [2] containing hourly recordings of a gas, multi-sensor device deployed in an Italian city from September 1 to October 31, 2004. The dataset contains five classes corresponding to different gases: *CO*, *NMHC* (non-metanic hydrocarbons), NO_x , NO_2 and O_3.

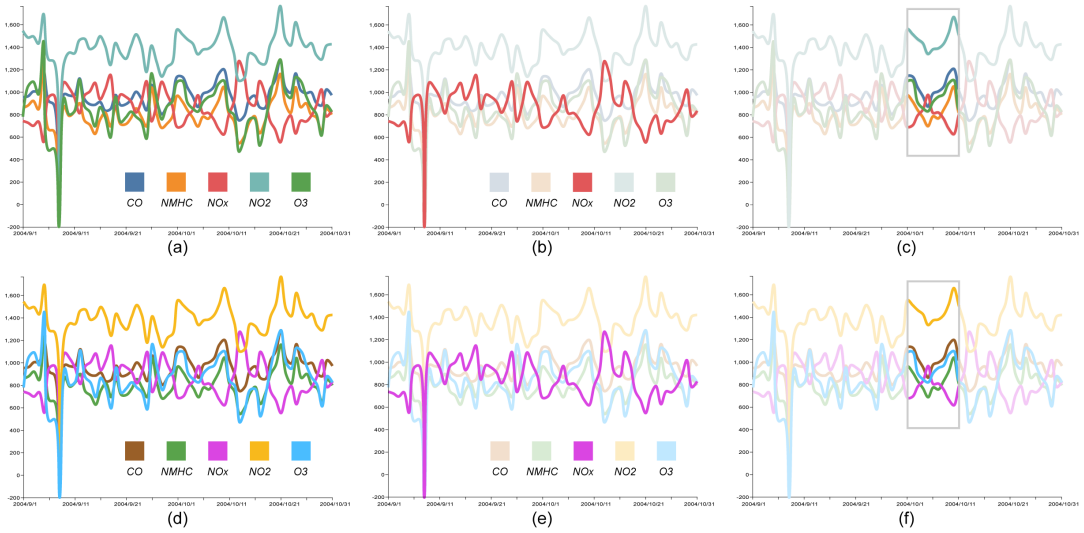

Fig. 13. Visualizing an air quality dataset [2] with Tableau Highlighter (a) to highlight the trendline for NO_x (b) and the brushing selection (c). The same purpose for our highlighting method (d, e, f). Our method (bottom row) gives a good highlighting effect while maintaining class discriminability during interactive exploration.

Figs. 13(a-f) shows line charts colorized using Tableau (top row) and our technique (bottom), where each gas type is represented using a unique color. We explore one class by interactively emphasizing it using Tableau's Highlighter. Fig. 13(b) emphasizes the red class, which represents NO_x. Here, the emphasis is good but the details of other classes are difficult to distinguish. By comparison, the class separability of our technique is not worse than Tableau for a baseline visualization (see Fig. 13(d)). With a focus on NO_x, our method achieves better overall class discriminability than Tableau (Fig. 10(e)), allowing the user to still investigate any de-emphasized classes. From the brushing section results shown in Figs. 13(c, f), we can see that our technique maintains better separability between all data points while Tableau Highlighter results in a few similar colors for non-selected data points. The ability to interactively vary the highlight while still maintaining context makes our method especially suited for visual exploration.

Compensation details.

The average spending time and the compensation for each task can be seen in Table.1. We also provided the experiment data and analysis code in the supplementary materials. You can recreate all the experiment results shown in our paper through the source files (*.Rmd).

Table 1. The average spending time and compensation for each task of the two crowd-sourcing experiments.

Task	Average Spending Time	Compensation
Counting Task	13.87min	$1.75
Selecting Task	16.80min	$2.00
Highlighting Task	4.12min	$1.00
Matching Task	6.90min	$1.25

REFERENCES

[1] Peter Bormann, Peder Aursand, Fahad Dilib, Surrender Manral, and Peter Dischington. 2020. *FORCE 2020 Well well log and lithofacies dataset for machine learning competition.* https://doi.org/10.5281/zenodo.4351156

[2] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia. 2008. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical* 129, 2 (2008), 750–757. https://doi.org/10.1016/j.snb.2007.09.060

[3] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[4] K. Lu, M. Feng, X. Chen, M. Sedlmair, O. Deussen, D. Lischinski, Z. Cheng, and Y. Wang. 2021. Palettailor: discriminable colorization for categorical data. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 475–484. https://doi.org/10.1109/TVCG.2020.3030406

[5] Tableau Software. [n.d.]. The tableau visualization system. http://www.tableausoftware.com/.

[6] Yunhai Wang, Xin Chen, Tong Ge, Chen Bao, Michael Sedlmair, Chi-Wing Fu, Oliver Deussen, and Baoquan Chen. 2019. Optimizing color assignment for perception of class separability in multiclass scatterplots. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 820–829. https://doi.org/10.1109/TVCG.2018.2864912