Deep Colormap Extraction from Visualizations

Supplementary Material

Lin-Ping Yuan, Wei Zeng, *Member, IEEE*, Siwei Fu, *Member, IEEE*, Zhiliang Zeng, Haotian Li, Chi-Wing Fu, *Member, IEEE*, and Huamin Qu *Member, IEEE*

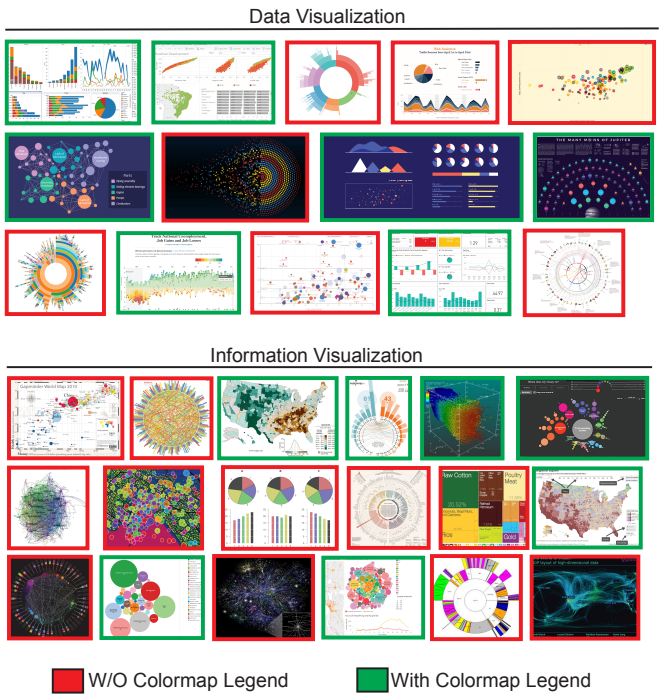


Fig. 1. Popular visualization images queried on Google Image with keywords 'data visualization' (top), and 'information visualization' (bottom). Over 50% of them do not have a color legend.

1 Background

Color is a vital visual channel that helps users effectively acquire information from visualizations. Most, if not all, visualizations employ a colormap, which is essentially a mapping function $f : D \rightarrow C$ that maps data values D to colors from the color scale C. To facilitate community, it is a good practice to incorporate a visualization with the colormap that produces it. Yet, many real-world visualizations do not come up with an explicit color legend. We queried on Google Image search engine with keywords 'data visualization', and 'information visualization'. The results are presented in Fig. 1, where visualizations with a color legend are marked green, and the others are marked red. Here, only 7/14 data visualizations, and 7/18 information visualizations have a color legend. Actually, queries with 'scientific visualization' produce even worse numbers. Without an explicit color legend, it would be rather challenging to transfer colormap design from anesthetic visualizations. This motivates us to develop an automatic approach to extract colormaps from visualization images. We consider a basic requirement for the tool is to work for visualizations with & without color legend.

2 Synthetic Visualization Corpus

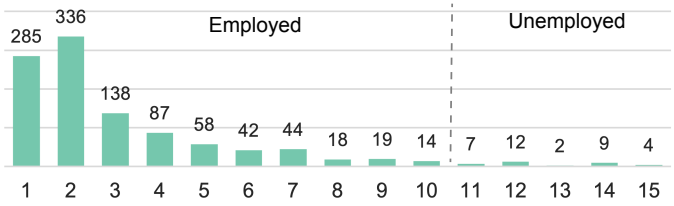


Fig. 2. Number of data distribution against number of attributes in the range [1, 15] in Rdatasets.

We consider the following three perspectives when synthesizing visualization images.

- *Data*. We utilize a subset of real-world data collected by Rdataset. Fig. 2 presents the number of data distribution against number of attributes in the range [1, 15] in Rdatasets. Here, we employ data with one or two attributes to cope with continuous colormaps, while those with 3 - 10 attributes for discrete colormaps. We omits data with more than 10 attributes as they are rare, and we find very few discrete colormaps with over 10 colors.
- *Chart type*. We consider these popular charts: {*line chart*, *pie chart*, *grouped bar chart*, *stacked bar chart*, *scatter plot*, *stream graph*, *heat map*, *choropleth map*}. *Heat map* and *choropleth map* are color coded with continuous colormaps, while the other are with discrete colormaps.
- *Colormap*. We collect in total 236 − 54 continuous and 182 discrete colormaps. Specifically, the number of discrete colormaps are 38, 38, 38, 38, 10, 10, 6, 4 for number of colors from 3 to 10; and the number of continuous colormaps are 2 for cyclical, 8 for single-hue, 22 for multi-hue, and 22 for diverging colors.

Fig. 3 presents an overview of number of synthetic charts produced by discrete colormaps of different color numbers. To ensure the model is unbiased by chart types, we keep balanced total number of charts for each chart type.

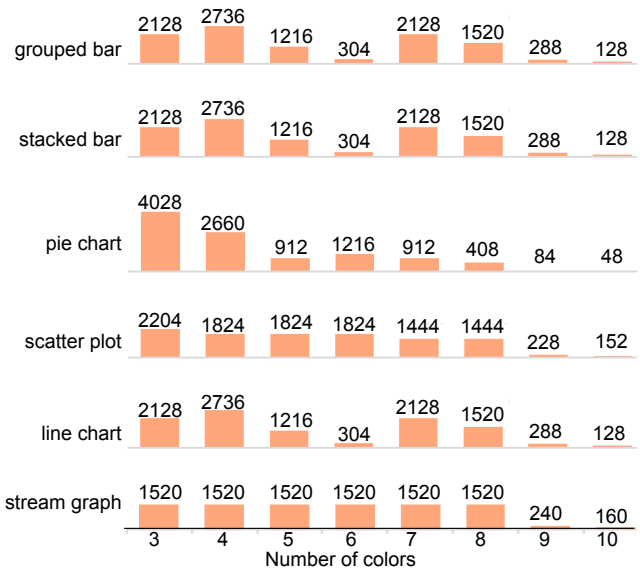


Fig. 3. Numbers of synthetic charts in 6 chart types with discrete colormaps of different color numbers.

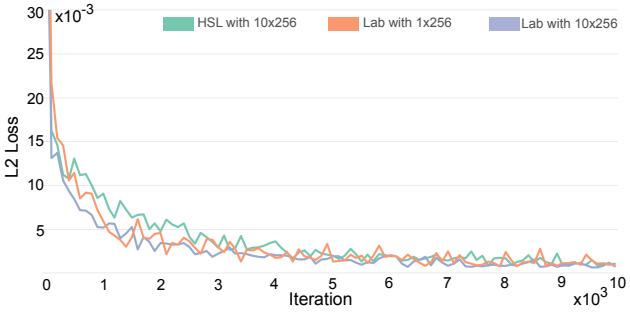


Fig. 4. Comparisons of network input as 2D map concatenation of HSL histogram (green) *vs.* Lab histogram and prediction colormaps of size 1×256 (orange) *vs.* Lab histogram and prediction colormaps of size 10×256 (blue).

3 Ablation Study

Ablation analysis. Besides the models presented in the paper, we train more deep neural network models of i) conversion to HSL instead of Lab color space, and ii) prediction colormaps in different output size of 1×256 instead of 10×256. All other hyperparameter settings employed in the networks are the same. Fig. 4 presents the comparison results. We can notice that the training loss are almost the same that eventually converge to around 0.50×10^{-3} after iteration 10,000. In consideration of perception uniformity, we select the Lab color space; to make more color samples for the refinement process, we opt to set output size as 10×256.

Regression or classification. In the paper, we train the CNN as a regression model whose output is a $10 \times 256 \times 3$ array and loss function is the mean squared error loss. An alternative approach is to frame the problem as a multi-class classification task and directly categorize input histograms into classes of colormaps. To compare these two options, we also train a classification CNN model with the same input, network architecture, and hyperparameters as the regression model. We change loss function to the cross-entropy loss, and the output to a 1×236 vector that indicates the probability of the input histogram being one of the 236 colormaps in the training dataset. We evaluate its performance on the synthetic testing and real seen visualizations, except for real unseen visualizations. The classification model achieves an accuracy of 94.98% and 34.13% on the synthetic testing and real seen visualizations, respectively.

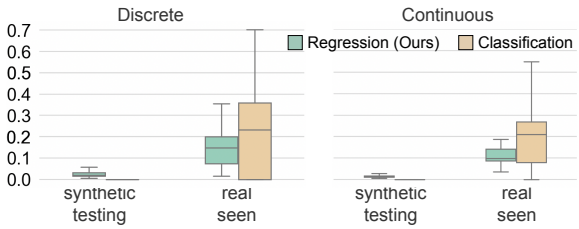


Fig. 5. Comparing D_{dtw} generated by the regression model (our method) and classification model. The regression model achieves smaller D_{dtw} for real visualizations.

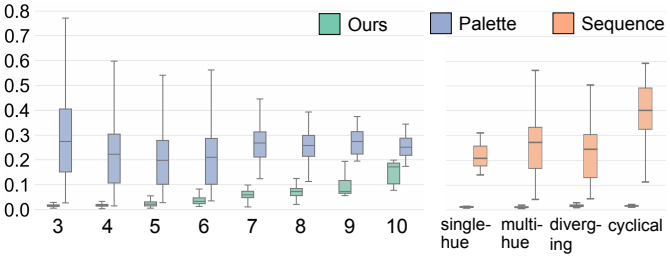


Fig. 6. Comparing D_{dtw} generated by Palette [1] on number of colors in discrete colormaps, Sequence [4] on categories of continuous colormaps, and Ours on all cases.

To further compare the regression and classification approaches, we calculate their DTW distances (denoted as D_{dtw}) between ground truths and extracted colormaps by each method. The results are shown in Fig. 5. Overall, the classification model performs better on synthetic testing visualizations, but it is inferior to the regression model on real seen visualizations. Specifically, both approaches produce satisfactory results for synthetic testing visualizations, yet the classification model is slightly more accurate and stable, especially for discrete colormaps. However, for real visualizations, the classification model has a significantly ($t = 2.99, p < 0.01$) larger D_{dtw} ($\mu = 0.23_{[0.19,0.27]}$) than that of the regression model ($\mu = 0.14_{[0.13,0.15]}$) for discrete colormaps, and a significantly ($t = 3.45, p < 0.001$) larger D_{dtw} ($\mu = 0.20_{[0.15,0.25]}$) than that of the regression model ($\mu = 0.11_{[0.10,0.12]}$) for continuous colormaps. After examining the classifier's predictions on real visualizations, we find that it often predicts a discrete colormap whilst the ground truth is a continuous colormap, and vice versa. The classifier sometimes even predicts a colormap with a different color scheme of that in the ground-truth colormap. A possible reason is that real-world visualizations feature noisy color histograms, which can be easily caused by many practices such as changing image resolutions and opacity. Moreover, the classification model can never categorize unseen colormaps that are not in the training dataset, such as those real unseen visualizations. In this sense, the regression model is more robust and generalizable. Hence, we adopt the regression model as our approach.

4 More Quantitative Experiments

4.1 Comparison with Palette-Based and Sequence-Preserving Methods

We further evaluate our method against palette-based [1] and sequence-preserving [4] methods by comparing their performances on number of colors in discrete colormaps, and different categories of continuous colormaps. The results are presented in Fig. 6. Overall, our method outperforms both palette-based [1] and sequence-preserving [4] methods significantly. Besides, we find

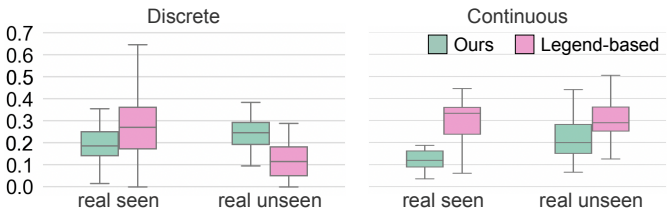


Fig. 7. Comparing D_{dtw} generated by our method and the legend-based [3] method on real-world visualizations with explicit legends.

that performance of our method drops when the number of colors increases for discrete colormaps, while palette-based [1] method almost remains the same. Through a deep probe, we observe that the network tends to predict discrete colormaps as continuous ones when the number of color increases; see examples in Fig. 8. For continuous colormaps, we notice that our method performs consistently across different colormap categories. In contrast, the performance of sequence-preserving [4] method reduces when the number of hues increases from single-hue to cyclical colormaps. This is because it is difficult for sequence-preserving [4] method to arrange colors in a correct ordering if there are many hues. We also find that it is easier for our network to predict continuous colormaps than discrete colormaps. Here, a possible reason is that in our colormap corpus, there are several discrete colormaps with the same color scheme (e.g., RdYlBu or BuGn) but different numbers of colors (e.g., 4 or 5). The network sometimes messes up these similar discrete colormaps for a visualization whose underlying data distribution is unbalanced.

4.2 Comparison with Legend-Based Method

Poco et al. [3] propose a five-stage pipeline to extract colormaps based on legends. We also conduct a quantitative comparison with this legend-based method [3] on a dataset of real-world visualizations with explicit legends. We first select such visualizations from our collected real seen and real unseen visualizations, obtaining 61 visualizations with seen colormaps and 87 with unseen colormaps. Then, following the practice in [3], we manually crop the legend regions for each visualization. Finally, we re-implement the color extraction method as described in the paper to recover colormaps from the legend regions.

We measure D_{dtw} between the extracted and ground-truth colormaps. The comparison result with our method is shown in Fig. 7. Here, our method performs equally or better than legend-based [3] method on the real seen dataset. On real unseen dataset, our method works better for continuous colormaps but worse for discrete colormaps. In more details,

- *Discrete colormaps.* Though no significance is observed between our and legend-based [3] method on real seen visualizations ($t = -0.95, p = 0.34$), our method reduces mean D_{dtw} from $0.26_{[0.16,0.35]}$ to $0.19_{[0.16,0.23]}$. We notice that the performance of the DBSCAN algorithm used in [3] is largely influenced by image resolution. Low-resolution visualizations can result in repeated or unexpected colors in the colormaps extracted by [3]. Our method alleviates this effect by utilizing a CNN model to predict an intermediate colormap before clustering, see Sec. 7.1 in the paper. However, our method has a larger D_{dtw} ($\mu = 0.26_{[0.22,0.31]}$) than that of [3] ($\mu = 0.11_{[0.05,0.17]}$), and the difference is significant ($t = 3.7, p = 0.001$) on real unseen colormaps. This is because the network has never seen some colors in the unseen colormaps, especially in some uncommon colormaps.
- *Continuous colormaps.* Our method outperforms the legend-based method [3] with mean D_{dtw} reduced from $0.33_{[0.28,0.38]}$ to $0.12_{[0.10,0.14]}$ on real seen dataset ($t = -6.0, p < 0.0001$), from $0.29_{[0.24,0.34]}$ to $0.20_{[0.17,0.23]}$ on real unseen dataset ($t = -2.5, p = 0.01$). Here, [3] recovers continuous color legends by flood-filling pixels within a black border. The algorithm can easily fail to detect the whole legend area if a black border does not exist or is not distinct from the background.

Moreover, our method has another advantage of requiring no explicit color legends, as the method works on the color histogram domain. On the other hand, the legend-based method [3] can extract more information including color values. In together, ours and [3] can complement each other in supporting more general cases.

5 Qualitative Examples

Fig. 8 presents more results produced by our CNN model from the synthetic dataset. The left column presents four good examples of discrete colormaps. The chart types cover scatter plot, bar chart, line chart, and stream graph, while the colormaps include single-hue, diverging, and categorical types. Notice that the line chart (Example 1-3) is encoded with grayscale discrete colormaps, and the CNN successfully predicts a good result that can be refined. The middle column presents four good examples of continuous colormaps. The results are very similar to the ground truth colormaps. Nevertheless, our method may fail, as demonstrated by the four examples in the right column. Here, we find that a main reason is unbalanced data distributions (Examples 3-1, 3-3, 3-4). We also notice that the network tends to predict continuous colormaps for single-hue discrete colormaps, especially when the colormap contains over 7 colors (Example 3-2).

Fig. 9 presents some results produced by our model for real-world visualizations with seen colormaps from the Internet and the study of Poco et al. [3] (highlighted with red boxes). Many predicted colormaps can be refined to generate the correct colormaps. Yet, some predictions are rather noisy (Example 1-4, 2-4), as the visualizations contain overlapping circles, making a big change in the color histogram that is not learned by our model.

Fig. 10 shows some CNN predictions for real-world visualizations with unseen colormaps. We notice that the network tends to predict a similar colormap from those in the training dataset. For example, prediction for Example 1-1 is in our dataset, but it misses some green colors in the ground truth. Some predictions only capture partial colors of the ground truth, e.g., Examples 1-2, 1-3, 1-4. This is probably because the underlying data are skewed, and some colors in the ground truth are not fully utilized. Some extracted colormaps are quite close to the ground truth with small hue differences, e.g., Examples 2-1, 2-2, 2-3, and 2-4. These ground-truth colormaps can be easily created by connecting certain color paths in the color space [2], but are not available in the libraries we harvested. The unseen discrete colormaps in Examples 3-1, 3-2, 3-3, and 3-4 use partial or additional colors of common discrete colormaps in our dataset. Our model produces noisy predictions, some of which can be corrected using the refinement module.

References

[1] H. Chang, O. Fried, Y. Liu, S. DiVerdi, and A. Finkelstein. Palette-based photo recoloring. *ACM Trans. Graph.*, 34(4):1–11, 2015.

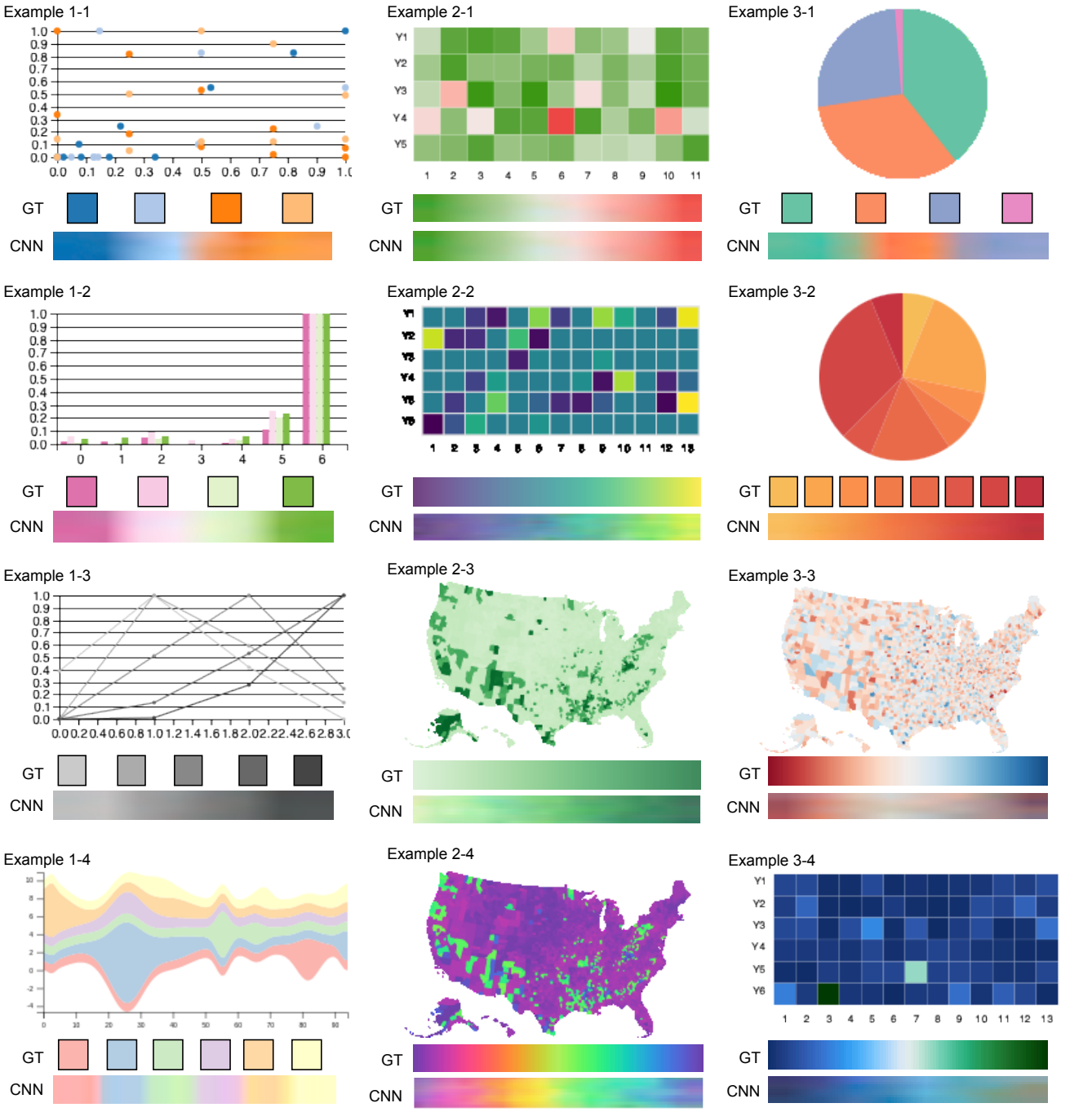


Fig. 8. Example colormap results extracted by our CNN model from the synthetic visualizations: good examples for discrete colormaps (left), good examples for continuous colormaps (middle), and bad examples for both discrete and continuous colormaps (right).

[2] P. Nardini, M. Chen, F. Samsel, R. Bujack, M. Bottinger, and G. Scheuermann. The making of continuous colormaps. *IEEE TVCG*, 2020. In print.
[3] J. Poco, A. Mayhua, and J. Heer. Extracting and retargeting color mappings from bitmap images of visualizations. *IEEE TVCG*, 24(1):637–646, 2018.
[4] M. J. Yoo, I. K. Lee, and S. Lee. Color sequence preserving decolorization. *Comput. Graph. Forum*, 34(2):373–383, 2015.

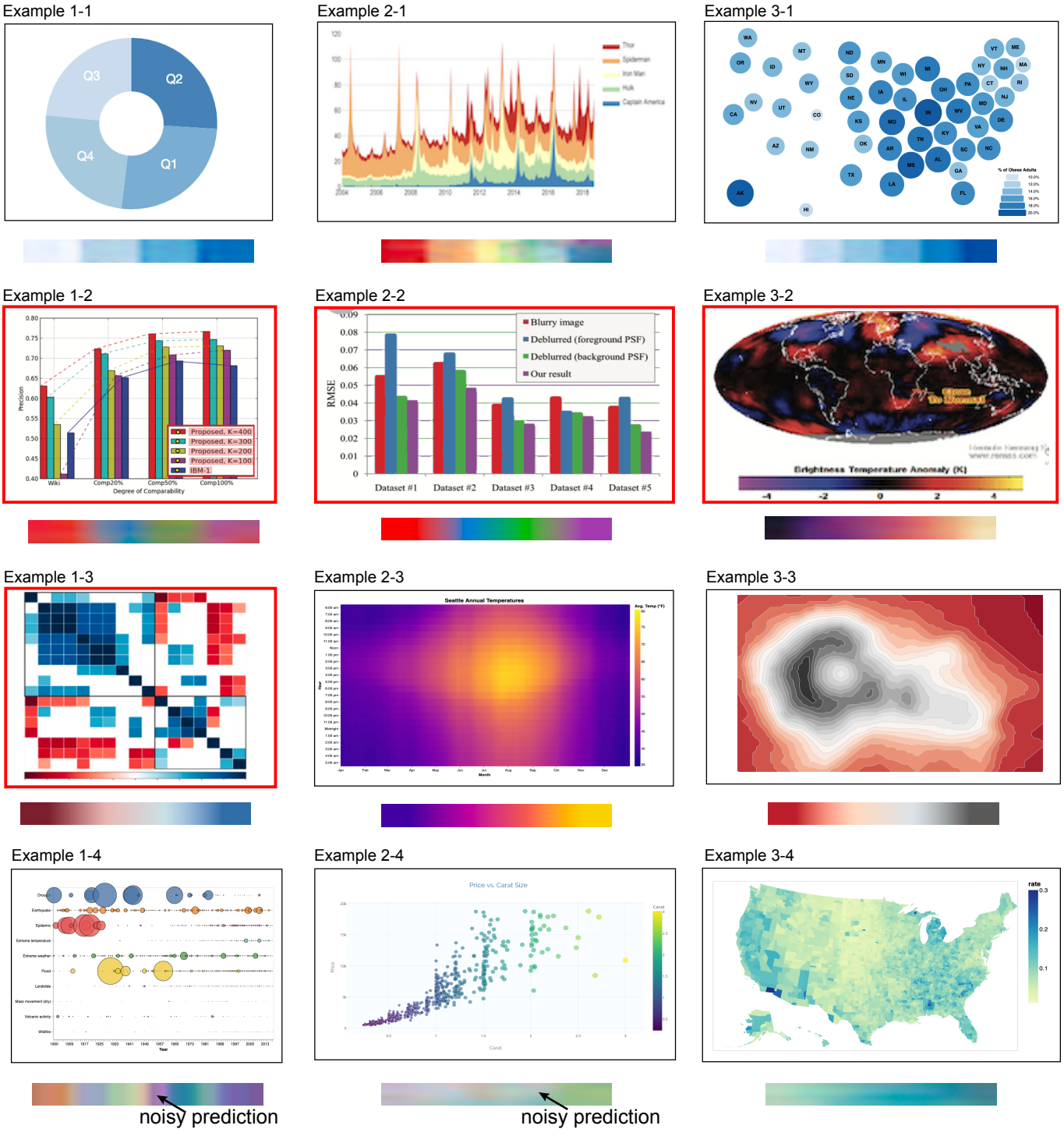
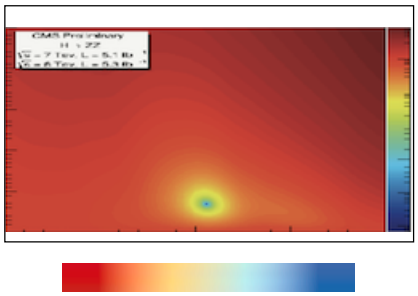
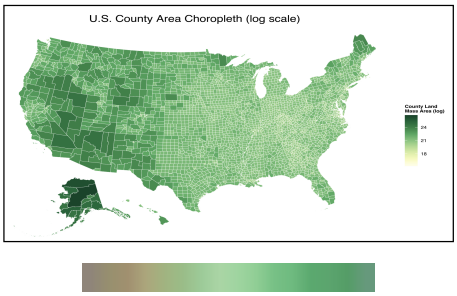


Fig. 9. Example colormap results extracted by our CNN model for the real-world visualizations with seen colormaps, collected either from the Internet or the prior study of Poco et al. [3] (highlighted with red boxes). Colormaps predicted by our CNN model are presented underneath the visualization images.

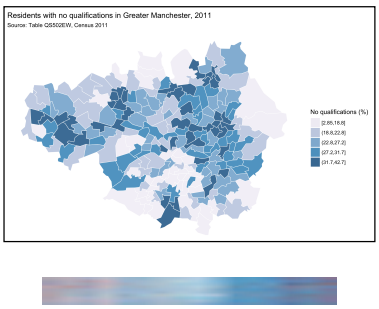
Example 1-1



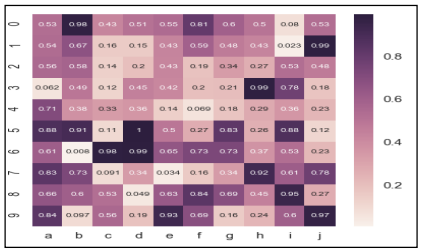
Example 2-1



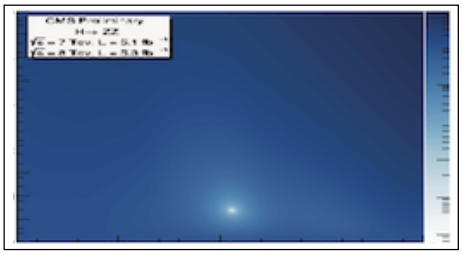
Example 3-1



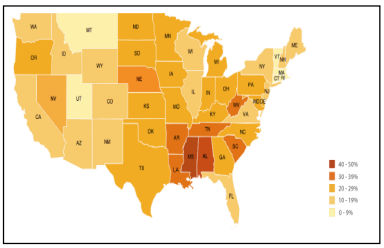
Example 1-2



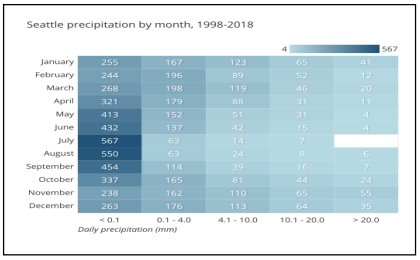
Example 2-2



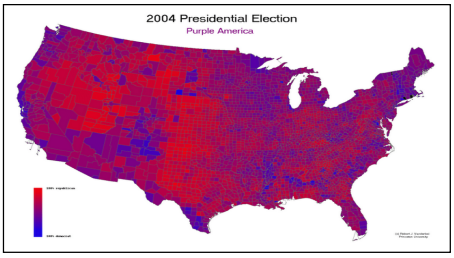
Example 3-2



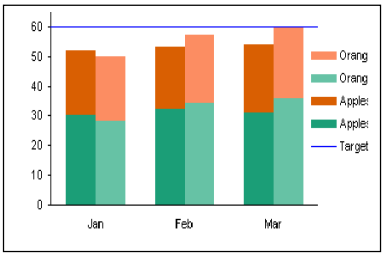
Example 1-3



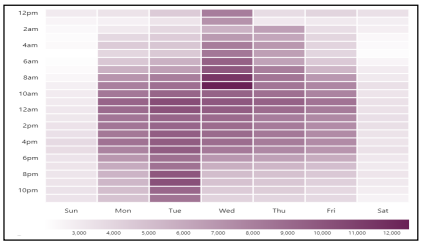
Example 2-3



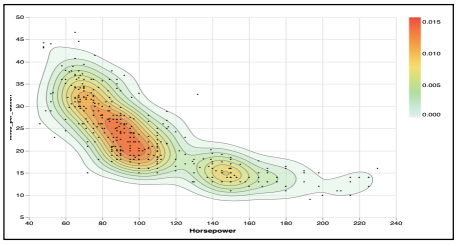
Example 3-3



Example 1-4



Example 2-4



Example 3-4

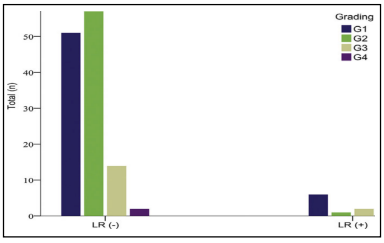


Fig. 10. Example colormap results extracted by our CNN model for the real-world visualizations with unseen colormaps. Colormaps predicted by our CNN model are presented underneath the visualization images.