Personalized sketch-based brushing in scatterplots

Chaoran Fan
University of Bergen

Helwig Hauser
University of Bergen

Brushing is at the heart of most modern visual analytics solutions and effective and efficient brushing is crucial for successful interactive data exploration and analysis. As the user plays a central role in brushing, several data-driven brushing tools have been designed that are based on predicting the user's brushing goal. All of these general brushing models learn the users' average brushing preference, which is not optimal for every single user. In this paper, we propose an innovative framework which offers the user opportunities to improve the brushing technique while using it. We realized this framework with a CNN-based brushing technique and the result shows that with additional data from a particular user, the model can be refined (better performance in terms of accuracy), eventually converging to a personalized model based on a moderate amount of retraining.

Nowadays, linking and brushing is becoming a prevalent interaction technique for data exploration and analysis in coordinated multiple views,[1] widely integrated into many visualization tools such as Tableau and D3.js. Becker and Cleveland[2] were the first to describe the basic principles of brushing, which is interactively painting the visualization of a subset of data points with usually simple geometries. In coordinated multiple views, the selected data are consistently highlighted in all linked views after brushing. This amounts to the most common form of focus+context visualization,[3] enabling a fast and effective exploration of data relations, which are too challenging to show in just one view.

Due to the importance and prevalence of brushing in visualization solutions, many efforts have been invested in the improvement of brushing technique and brushing variants can be categorized into:

- **brushing using simple geometries**—the most commonly used brushing solutions include the rectangular or circular brushing on scatterplots, line-brushing on data graphs, etc.
- **lassoing**—the user draws a detailed shape around the target group of item representations to select the subset
- **logical combinations of simple brushes**—the user refines the data selection by making use of multiple brushes and combining them with logical operators[4-5]
- **sketch-based brushing**—a sketch is drawn by the user onto a visualization and a selection heuristic is used to determine which data are selected[6-8]

Each brushing technique can be discussed in terms of pros and cons, especially by two particularly important criteria:

- **efficiency**—whether the brushing is fast enough (including the interaction and all computation) to enable a fluid data exploration/analysis[9]
- **accuracy**—does the brushing interaction lead to a selection of exactly the data subset, which the user wished to select in the view?

Despite the large variation of existing brushing tools, we rarely see a solution that combines both criteria really well: Many brushing techniques are indeed fast, as clicking on individual points, for example, or drawing simple geometries—also sketched brushes are fast, requiring only a simple gesture as interaction and thus enabling a swift user-computer dialogue during the exploration/analysis.[10] A common disadvantage of fast techniques, however, is that it can be difficult to accurately brush a particular data subset. On the other hand, we certainly find brushing techniques, that are straight-forward for accurately selecting subsets of interest, such as lassoing and the logical combination of simple brushes. This benefit, however, comes often at the price of the solution being slower—specifying a lasso, for example, easily becomes a unit task by itself,[10] potentially interrupting the exploration/analysis process.

Recently, deep learning methods have become the state-of-the-art in a wide range of fields, including natural language processing, object detection and image classification. Inspired by this, we have recently developed a CNN-based brushing technique for scatterplots and achieved the best accuracy compared to previously existing sketch-based methods with a fast interaction.[11] Although this method has made great progress in learning the user's intention while brushing, it suffers from two limitations: First, the model is trained off-line once by the data from different users and what the model learns is the average brushing preference across the users, leading to a general model which is obviously not optimized to every single user. Second, while it of course is possible to retrain a new model for a single user from scratch, this procedure is time-consuming and requires sufficient training data which is difficult to get from a single user in a short time.

To address the limitations of this CNN-based brushing solution, we here propose an innovative framework which is able to iteratively refine the brushing model for a single user with additional data that he/she provides while using the brushing technique. This idea is inspired by active learning (AL), which is a special case of semi-supervised machine learning that can incrementally improve the existing model by interactively querying the user for additional input. In addition, we exploit knowledge from transfer learning and leverage the parameterization of a well-trained model instead of learning the user's brushing behavior from scratch, largely reducing the time cost of the retraining procedure while maintaining a focus on avoiding overfitting.

To evaluate the usability of our proposed framework, we experimented with the previously published CNN-based brush in scatterplots to see relevant differences in comparison. Our quantitative evaluation shows that the iterative model based on our proposed framework achieves better accuracy and becomes a customized model for a single user as compared with the general model, and the time cost for the retraining procedure is only 3 minutes, which is efficient and acceptable for most data analysts to improve their brushing method even during a short break only.

RELATED WORK

In the following, we first review some critical works concerning brushing for visual analytics, before we then discuss related work concerning applications of convolutional neural networks, transfer learning and active learning.

Brushing techniques

Brushing is intrinsically based on the interaction between the user and the system, often a combination of mouse/cursor motions and clicks. Many variations of brushing have been proposed, each with its own strengths and weaknesses—for example, in terms of their ease of use and the degree of control that the user has. Brushing in scatterplots is often based on the use of simple geometric shapes such as a rectangle or circle. Alternatively, users can use a lasso to specify the selection more accurately.

Several extensions to simple brushing have been published, including techniques to formulate more complex brushes by combining multiple brushes using logical operators. Martin and Ward[4], for example, enable the user to configure composite brushes by applying logical combinations of brushes, including unions, intersections, negations, and exclusive or operations.

Similarity brushing[12] is a typical example of sketch-based brushing, which is based on a fast and simple sketching interaction—the user uses a swift and approximate gesture (for example, drawing an approximate shape that the data should follow) and then a similarity measure (target function) is defined to identify, which data items actually are brushed. This way, the interaction is fast, but likely not 100% accurate.

Recently, the Mahalanobis brush was presented as an interesting alternative for brushing scatterplots.[8] The user simply clicks into the center of a coherent data subset to be selected. The link between the interaction and the actual selection is realized on the basis of an analysis of the underlying data (a local covariance matrix indicates the overall shape and orientation of the data to be brushed, forming the basis for a local Mahalanobis metric, which is then used as a distance measure to select the data).

While this technique is giving quite good results, it still has limitations, including a non-optimized selection of the local context for the Mahalanobis computation and one off-screen parameter for the brush size. As a follow-up work, we extended the Mahalanobis brush and improved the accuracy by optimizing the parameters based on a user study and getting rid of the off-line parameter.[7] However, this improved solution is still linear and has difficulties with complex structures that would require a more flexible approach. Also, it is not really real-time for large datasets.

Later, we exploited deep learning and developed a CNN-based brushing in scatterplots.[11] This model achieves state-of-art accuracy while—as a general model—it only learns the average behavior from different users, and thus is not able to match every single user's brushing preferences in an optimal way.

Koytek et al.[13] created MyBrush, which extended the popular brushing and linking technique by incorporating personal agency. It offers users the flexibility to configure the source, link, and target of multiple brushes, which is able to adapt brushing and linking to preferences and needs. However, the user needs to spend some time to figure out all the possibilities of the configuration initially, and the brushing tool they provide is not based on a data-driven method which cannot be improved while being used.

CNN, transfer learning and active learning

A convolutional neural network (CNN) is a deep learning architecture, which is inspired by the connectivity pattern between neurons and their organization in the visual cortex.[14] The concept of a neocognitron, proposed by Fukushima,[15] is widely considered as fundamental basis of modern CNNs. LeCun et al.[16] established the framework of CNNs by developing a multi-layer artificial neural network called LeNet-5, which was applied successfully to image classification problems. With the emergence of big data and the development of computing infrastructure, the structure of some CNNs has become very deep. A solution by Krizhevsky et al.[17] was able to classify about 1.2 million images into 1000 classes, i.e., a record-breaking result in the ImageNet Large Scale Visual Recognition Challenge. Often, the impressive success of image processing CNNs is attributed to their ability to learn rich mid-level image patterns as opposed to hand-designed low-level features used in more traditional methods.

As humans we can learn and apply relevant knowledge from previous learning when encountering new tasks. Most of traditional machine learning algorithms are designed to address single tasks. In contrast, transfer learning allows us to bring the power of state-of-the-art models to new domains where insufficient data and time/cost constraints might otherwise prevent their use.[18] Transfer learning with CNNs has been also explored and demonstrates that the intermediate activations learned with pretrained deep CNNs on large datasets such as ImageNet and GoogLeNet, can be transferred to many other recognition tasks with limited training data.

Active learning (AL) is a special type of semi-supervised machine learning which incorporates the user into the loop to query label information. As labeling manually is expensive and time-consuming, AL has been successfully applied to the situations where large portions of data are unlabeled. The goal of AL is to improve the training performance of a classifier at the lowest possible annotation cost by intelligently picking the best examples to label.[19]

While an increasing number of successful applications of deep learning methods are recognized in many fields, we do not yet see a mentionable number of visualization solutions, particularly in improving interaction techniques. With our work, benefitting from ML, we also hope to inspire interesting new research in this very interesting direction.

THE PRINCIPAL APPROACH

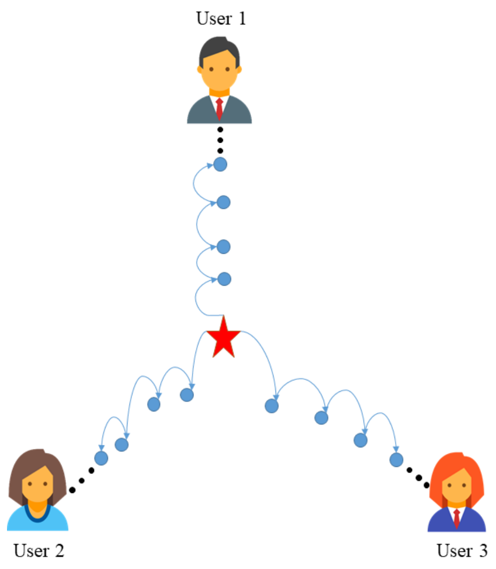

Figure 1. Illustration of the core idea: the general model (red star) is progressively refined and getting closer and closer to a single user model in the high dimensional parameter space (here 2D in this illustration).

The overall goal of our research was to improve the general CNN-based brushing model[11] and to make it suitable for those users whose brushing preference is deviating from the average. The schematic in Figure 1 shows the core idea of our proposed framework where the general model is indicated by a red star. To find the optimized model for a single user, we take advantage of the well-trained general model as an initial point in the model's high dimensional parameter space and progressively adapt it to a personalized model. In this way, the prior parameterization is taken into account and we can get rid of the costly general (global) search in the parameter space, largely reducing the time spent and making the results more stable.

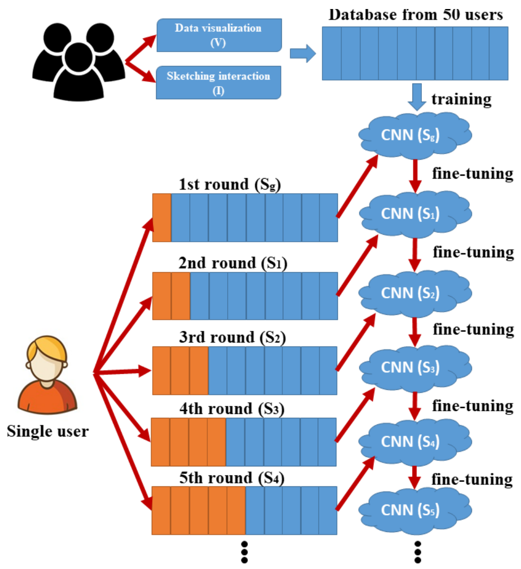

Figure 2. Principal approach: the general model is fine-tuned during continued use by new data from a specific user.

Figure 2 provides an overview of the proposed framework to illustrate our adaptive brushing model. In the following, we first describe the initial model we used from the earlier published paper,[11] before we then describe the construction of our solution in detail.

Following the basic definition of the general CNN-based brushing technique[11], we use click-and-drag as the sketching interaction **I** and a computational function for estimating the brushing result, denoted as **S**, which is based on the visualization **V** near sketch **I**. The CNN structure is built with two convolutional layers, two max pooling layers and three fully connected layers. The input of the CNN is a histogram-like image that is based on the interaction **I** and the local visualization **V** near **I**, while the output of the network is also an image, of same resolution as the input, which contains a degree-of-selection information extracted from the ground truth (users' brushing goal). The general model $\mathbf{S_g}$ is trained off-line once based on 500 basic selections (augmented to 8000 in actual training via sampling the natural variation of user interactions), provided by the previously published user study where 50 different users were invited.[11] In the user study,[11] a particular scatterplot (one out of six) and a particular request (one out of three) were given, and the participant chose a target data subset to select as ground truth by using a lasso tool, and then also provided the corresponding click-and-drag interaction that this participant would use to select the target group.

In our proposed framework, we establish an interactive scenario to improve the brushing accuracy for a single analyst during data exploration, where the user can actively give some new input when he/she does not like the result generated by the current brushing model. This additional data is used to gradually adapt the general model to a personalized one and we see that with more data from the user, the brushing result is more accurate for this user. To evaluate this framework, we organized a user study where single users were asked to participate in the model refinement procedure 5 times, and each time the user provided 10% new data by brushing new datasets. Then this user's personal brushing preference data contributed to updating the current model in retraining. The design of the user study intended to simulate a daily process of the data analyst while using the brushing tool during 5 working days. The time spent for participating in one round of the user study was around 20mins, which was comfortable for most users. We saw that 10% new data could be obtained in this time period and this amount of new data was reasonably enough to see the incremental effect of the retraining.

For learning a single user's brushing preference, we chose to leverage the existing CNN parametrization of the general model instead of retraining a model from scratch. This is because the data used for training the general model is from 50 different users and it is not practical to get a similar size data from a single user. To avoid overfitting by limited new data, the new data is chosen to replace the most similar data in the original dataset, composing a new training dataset rather than treating the additional data individually. Picking the most similar cases to be replaced is based on two reasons: 1. We aimed at the retrained model to learn the user's specific brushing preference and also keep itself as general as possible (with respect to cases not yet seen by the new user) at the same time. Therefore, the similar cases from the original data can be considered as replicated data which should be replaced. 2. Instead of adding the new data to the retraining dataset, the data replacement strategy accelerates the convergence during retraining.

Our retraining procedure is also based on transfer learning. Transfer learning strategies depend on various factors, but the two most important ones are the size of the new dataset (relatively small or big), and its similarity to the original dataset. As we replace old data, the new training data is the same size and of high similarity (90%) to the old training data. Therefore, we can fine-tune the weights of the pretrained current network via backpropagation with less of a chance to be overfitted. In the following, we introduce the algorithm for the dataset replacement in detail.

Training dataset replacement

The original data for training the CNN-based brushing model from our previous work[11] is based on 500 selections and the training data can be defined as $(\mathbf{x_i} \in \mathbf{T_{in}}, \mathbf{y_i} \in \mathbf{T_{out}})_{i=1}^{N}$ with pairs of input and expected output (N is the number of the training samples). In order to find the appropriate data to be replaced among the 500, we formulate a similarity metric based on the extracted features of the input data.

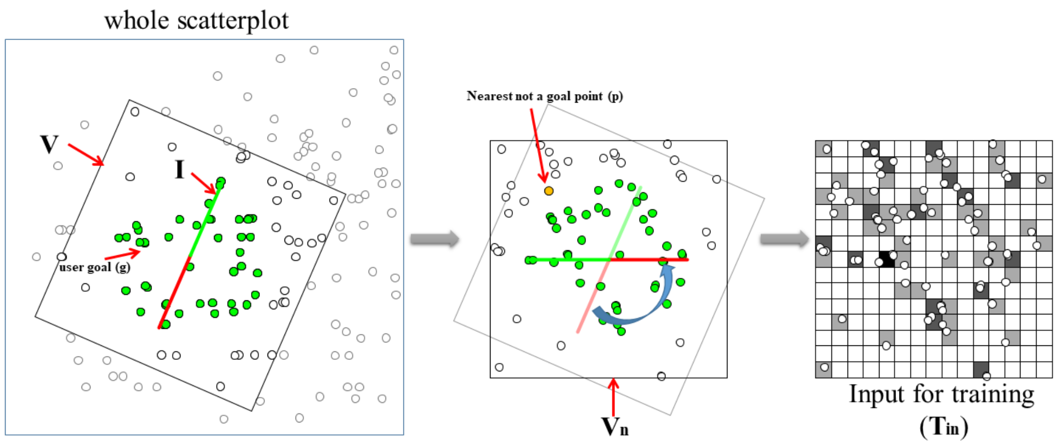

Figure 3. Illustration of CNN input computation: Left, a square area is specified by the interaction (red line segment); Middle, after rotating to the horizontal; Right, histogram of the local data distribution (CNN input).

Figure 3 shows the procedure of CNN input generation. The histogram-like image data used as input for training the network is converted from the local data visualization **V** and the related interaction **I**, the features we compute are mostly based on the local visualization **V** after normalization $\mathbf{V_n}$ with respect to the interaction **I** by a linear transformation. Moreover, to represent the features, we have the number of points within $\mathbf{V_n}$ denoted as n_v and the number of user goal points and all points in scatterplot are denoted as n_a and n_c respectively. Besides, the area of $\mathbf{V_n}$ is represented as s_v while the area of the scatterplot canvas is s_c.

To extract important and meaningful features from $\mathbf{V_n}$, the size of the user's brushing goal is obviously needed to be considered. We measure the influence of user goal size by calculating the ratios of the user goal size to all the points within $\mathbf{V_n}$ and the user goal size to all the points in the scatterplot, which can be denoted as $f_{pR} = \frac{n_g}{n_v}$ and $f_{gR} = \frac{n_g}{n_c}$ respectively.

To know whether the user focuses on a small or large area, we can compute the ratio of areas of $\mathbf{V_n}$ to the whole canvas of the scatterplot. This can be denoted by $f_{aR} = \frac{s_v}{s_c}$.

The geometric shape of the user goal is another important feature which needs to be taken into account. We estimate this by finding the nearest point (denoted by **p**, not a user goal) to the start point (denoted by **c**) of **I**, then compute the ratio $f_{nR} = \frac{d_E(\mathbf{p},\mathbf{c})}{r}$, where $d_E(\mathbf{p}, \mathbf{c})$ is the Euclidean distance between **p** and **c**, and *r* is the length of **I**.

As we know, the eigenvectors indicate the direction of points which are stretched by the transformation and the corresponding eigenvalue is the factor by which it is stretched. Therefore, the ratio of the eigenvalues is a proper way to measure the isotropy or anisotropy of the user goal, which can be defined as $f_{eR} = \frac{e_1}{e_2}, e_1 < e_2$, where the eigenvalues e_1 and e_2 are computed by eigendecomposition of the covariance matrix of the user goal.

In addition, we compute scagnostics to characterize the $\mathbf{V_n}$. Scagnostics is short for Scatterplot Diagnostics, first mentioned by John and Paul Tukey[20] to identify interesting structures according to density, skewness, shape, outliers, etc. The nine Scagnostics measures are defined as $f_{outlier}$, f_{skewed}, f_{clumpy}, f_{sparse}, $f_{striated}$, f_{convex}, f_{skinny}, $f_{stringy}$ and $f_{monotonic}$ respectively, and we use $\mathbf{f_{sg}}$ as shorthand of these 9 features together.

Lastly, we look at the density of the histogram-like image ($\mathbf{T_{in}}$) as it represents the data distribution. The value of each bin in $\mathbf{T_{in}}$ is normalized into [0, 1], we compute 4 indicators by counting the percentage of the value of each bin in [0, 0.25] (f_{q_1}), (0.25, 0.5] (f_{q_2}), (0.5, 0.75] (f_{q_3}) and (0.75, 1] (f_{q_4}), respectively.

Overall, we use 18 different indicators to represent the user's selection. To preprocess the indicators for an efficient comparison, we use principal component analysis (PCA) to reduce the dimension from 18 to 8 while keeping 92% variance of the original dataset. The dissimilarity function which is used to pick the most similar case among the old data, can be defined as

$$dis(\mathbf{i_{new}}, \mathbf{i_n}) = d_E(f_{pca}(\mathbf{i_{new}}), f_{pca}(\mathbf{i_n}))$$

where $\mathbf{i_{new}}$ and $\mathbf{i_n}$ are the original features of new data and old data, respectively. d_E represents the Euclidean distance and f_{pca} are the features after PCA, $n \in [1,500]$. The most similar case has the minimum value in terms of the dissimilarity function.

RETRAINING THE CNN

Usually, high accuracy cannot be achieved unless enough training samples are provided. As we replaced the old data by the new data, so the selection samples in the retraining data are still 500, which are not enough for retraining. Therefore, we synthesize 7500 additional training data from the already acquired training set based on the user's natural interaction variation. This data augmentation method is illustrated in our previously published paper.[11] Then we optimize the parameters of the network based on the training data (N=8000) using the mean-squared error as a loss function. In the following, we will illustrate the details of retraining the CNN.

Training details

In practice, it is rare to train an entire CNN from scratch with random initialization. This is because it is relatively difficult to have a dataset of sufficient size that is required for the depth of the network. Therefore, it is very often still beneficial to initialize with weights from a pretrained model.

Since the initial model is quite general and capturing the common brushing preference, we choose to fine-tune the network to make the initial model adapt well to the target dataset. As the CNN features are more generic in early layers (such as edges or local shapes) and later layers of the CNN become progressively more specific to the details of the information contained in the target dataset, an effective way for fine-tuning the pretrained parameters is to make the learning process happen only in the fully-connected layers, and keep the parameters of the convolutional layers frozen. To validate this approach, we also recomputed the brushing accuracy based on retraining the whole network and found that the results were worse (about 5%) than freezing the early layers of the network, which supports that our retraining strategy is appropriate.

In our user study, we iteratively retrain the model based on the pretrained model 5 times. We start the retraining after the first round of the user study, where we used the general model from our previously published paper[11] as a starting point for optimization towards to single user's tailored model.

For the experiment, we retrained the network in Keras with Tensorflow as the backend, which provides useful GPU acceleration and coding flexibility. For the training and testing, we used a PC with an Intel Xeon E5-1650 CPU and an NVIDIA GeForce GTX 1080 GPU. During fine-tuning, as we expect the weights of the pretrained CNN to be relatively good, the learning rate was set to be 5×10^{-4}, which is half of the learning rate for training the initial model. In this way, the model will not be distorted too quickly and too much. And as we can make use of the existing parameterization, thus we only need 2000 epochs instead of 10000 (training from scratch) to obtain a good convergence towards a high-quality state. As a result, the retraining procedure for fine-tuning a model once only takes 3 minutes.

USER STUDY

As users have their own brushing preference, it is important to explore how the user uses our brushing tool and test whether it is possible to allow the model to be customized. To evaluate our proposed framework with the CNN-based brushing for a single user, we conducted a user study to collect the user's brushing data and then retrained the model to better fit a single user. In our user study, 8 users are invited who are students or employees (7 are from Delft University of Technology and 1 is from University of Bergen).

Study datasets

During the user study, we provided 25 different datasets for the user to brush, which all were not used for constructing the general model[11]. Besides, the users were encouraged to bring their own datasets to replace the provided datasets.

Study process

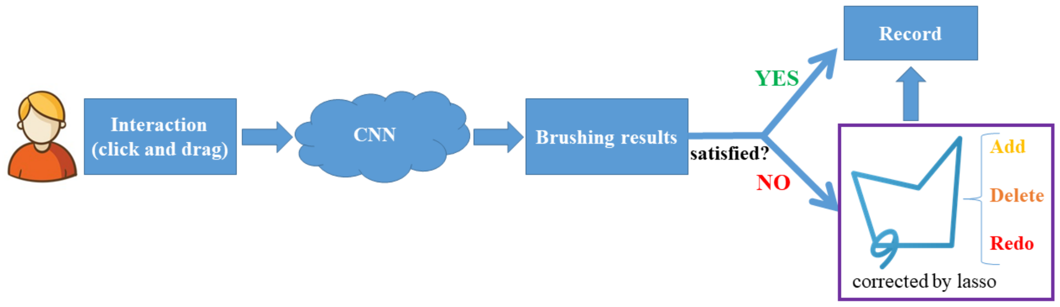

Figure 4. Pipeline of the user study. The user brushes a scatterplot based on the trained CNN model, and if he/she is not satisfied with the results, a lasso is offered to do corrections. In the end, the user's brushing goal, interaction and the brushing result are recorded.

Our user study consisted of two parts:

In the first part, a scatterplot was provided to the users and then they could freely use the click-and-drag interaction to brush some data subset of their choice.

In the second part, the brushing result based on the current CNN model was shown to the users immediately after they finished the interaction. Then they could think about whether the results were what they wanted originally. If not, they could use a lasso to do corrections (add and delete points, or directly specify the goal) until they were satisfied with the results. For a specific scatterplot, every user had to do 5 selections including the correction, if needed, before they would click the "next" button to switch to another scatterplot.

In order to investigate the influence of the retrained model based on the user's new brushing data, we asked each user to participate in the user study for 5 rounds and each scatterplot from 25 datasets showed up twice in total but in different rounds. For each round, the user needed to finish 5 selections in 10 different scatterplots, and these data were then applied to retrain a new, adapted brushing model which was then used for the next round. In the first round of the study, we directly used the already trained model from our previously published paper[11] as the initial brushing model. Then this model will be retrained by the data obtained from the first round of the study.

As the retraining procedure takes approximately 3 minutes, the users finished a full user study (5 rounds) in 5 discontinuous time periods. During the user study, the brushing results generated by the current model and the user's real goal and interaction were recorded. The pipeline of the user study is shown in Figure 4.

Before the start of the user study (only for the first time when the user joins), we presented our brushing technique in a training session, where we showed how the technique works and the interface operation in a test dataset. This session took approximately 10 minutes for each participant and the participants were free to interrupt for questions and to take over the software to experiment with the brushing technique (e.g. the click-and-drag interaction and correcting the results by lasso) until they felt ready to do the study.

Evaluation

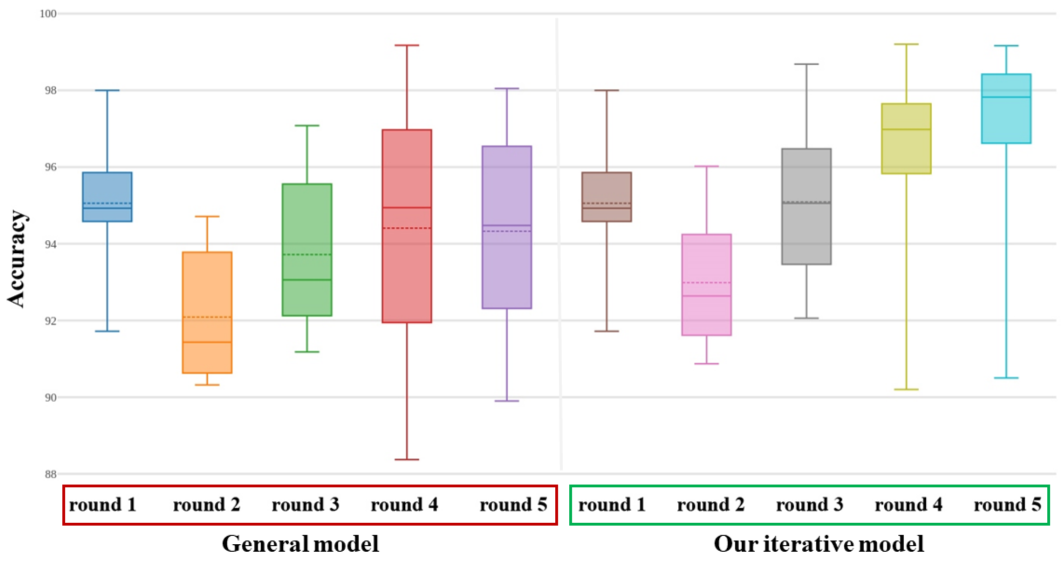

Figure 5. Boxplots of the brushing accuracy over rounds based on the general model (left) and our iterative model (right). The dash line and solid line in the box show the mean and median respectively.

For evaluating our proposed iterative model, we did a quantitative accuracy comparison with the previously published general model[11]. Figure 5 is a statistical comparison with boxplots. The left five boxplots and right five boxplots show the brushing accuracy of 8 users in all five rounds based on the general model (left) and our iterative model (right) respectively. The accuracy of our iterative model in each round is calculated based on the retrained model from the last round. In the first round of the user study, the user's brushing results are computed by the general model without optimization based on the new data, thus the difference between the general model and our iterative model appears from the 2nd round to the 5th round. The dashed line in the box is the mean, if we compare the accuracy by pairs in terms of the round, we can see after iterative retraining, our model performs better than the general one (with higher median and mean). Besides, the size of the boxes on the right are smaller than the corresponding one on the left, this indicates our model has less variance and becomes more stable.

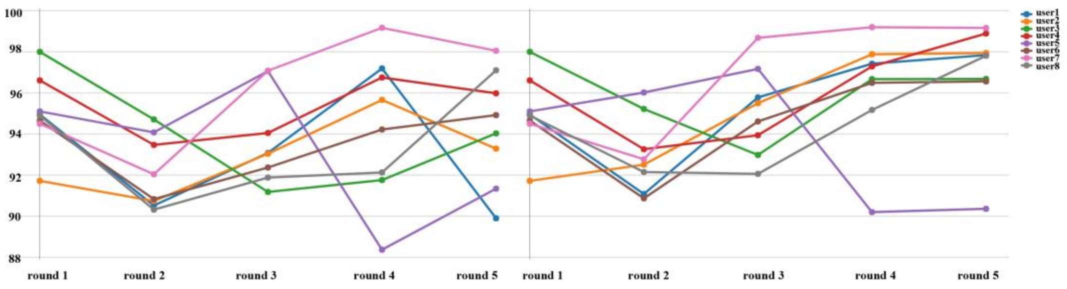

Figure 6. Line plots of the brushing accuracy over 5 rounds of 8 users based on the general model (left) and our iterative model (right).

Figure 6 shows a side by side accuracy comparison with line plots, between the general model (left) and our iterative model (right). By looking at the accuracy variation altogether, we can obviously see a rising trend of accuracy over rounds on the right side while the accuracy of the general model is more distributed (no special trend except a decline from round 1 to round 2) on the left. This shows a strong indication that with additional data for retraining the model, it is able to gradually learn the brushing preference of a single user, which leads to a more personalized model from the general model. In addition, we also see an outlier (user 5) with a sharp accuracy decrease in the last two rounds. To investigate the reasons behind, we pick the 6 worst performing cases from user 5 in the 4th and 5th round and list them in Figure 7. The brushing results are compared to the user goal (encoded by color). The true positives (correctly brushed), true negatives (correctly not brushed), false positives (falsely brushed) and the false negatives (falsely left out) are colored in yellow, white, pink and purple, accordingly. We can see that the user tried to select a complicated spiral shape and this kind of selection does not exist in previous rounds, being almost impossible for the network to predict at first time. For the relatively low accuracy (actually 90.36% is still very good) in round 5, as there are only 3 spiral-like cases from round 4 which contribute to the network retraining, so it is difficult to tweak the model to learn it.

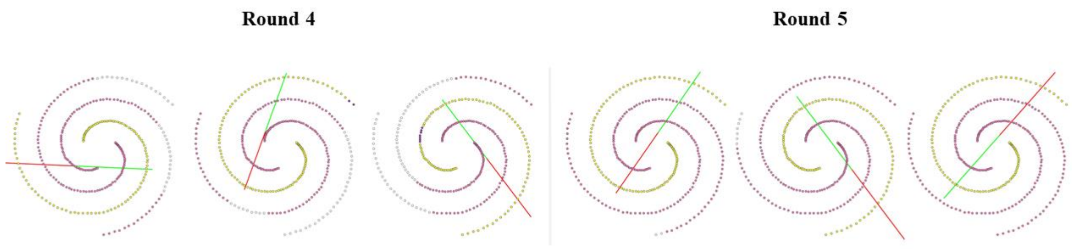

Figure 7. Worst performing cases of user 5 in round 4 (left) and round 5 (right).

During the retraining procedure in each round, the parameters of the network are updated. To understand what the network learns for each user and the relation between different user models, we extract the parameters of the CNN model in each round of each user, composing a vector with 35233 dimensions, which is denoted as $\mathbf{v}_{\mathbf{mn}}$, where *m* is the user ID and *n* is round number. In Figure 8, we compute the angle and absolute length differences to measure the similarities between these 40 different vectors, which are shown in two matrices with green (low value) to red (high value) as the color legend. In both matrix A and B, we see the outlier model in round 4 ($\mathbf{v}_{\mathbf{54}}$) and round 5 ($\mathbf{v}_{\mathbf{55}}$) of user 5 also have big difference with other users in this comparison. In addition, in matrix A, the angle between intra-user vectors are clearly smaller than inter-user's—this indicates that the model indeed approaches different user point in the parameter space. For matrix B, we see a clear pattern that the diagonals right next to the main diagonal are also very close to 0. This gives a strong impression that the model indeed adapts in small steps over the five rounds. In summary, the visualization of the CNN model parameters shows clear evidence that our iterative model learns reasonable parameters while being adapted instead of random search in the parameter space.

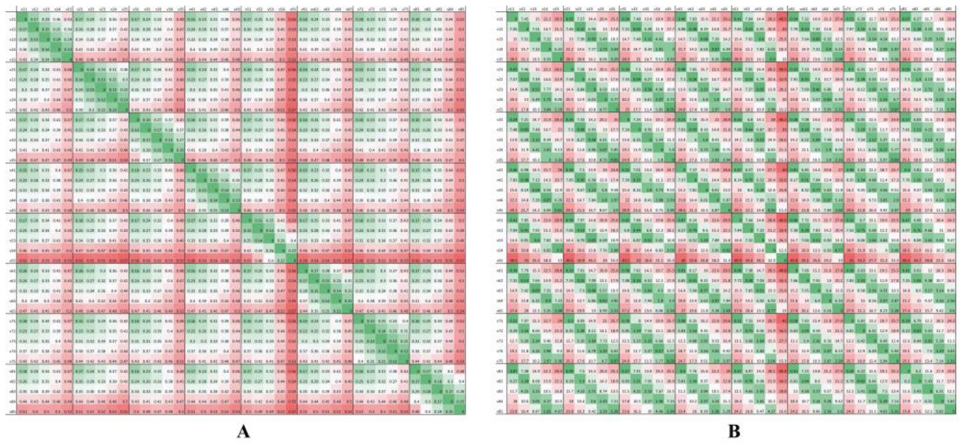

Figure 8. Vector similarity comparison of the CNN model parameters in each round of each user. Matrix A: angle between vectors. Matrix B: absolute length differences.

CONCLUSION AND FUTURE WORK

In this paper, we present a user-centric framework which takes the user in the loop to iteratively improve a brushing technique in scatterplots. By refining a general model for estimating data selections from simple click-and-drag interactions incrementally with the additional data from a single user and leveraging the existing parameterization, we achieve a solution which is able to turn the general model based on people's average brushing preference to a tailored model for the specific user with a very short time training cost ($\approx$3mins). In addition, we examined the quantitative performance of our iterative model in comparison to the general model, with the retraining procedure over time, the results show a clear improving trend compared to the general model: from 92.09% to 92.99% (+0.9%, with 10% new data), 93.72% to 95.09% (+1.37%, with 20% new data), 94.41% to 96.29% (+1.88%, with 30% new data) and 94.33% to 96.92% (+2.59%, with 40% new data). We assume the accuracy improvement can be even better with more user input and the most practical thing is that the retraining time cost is reduced to $\approx$4% of training a general model from scratch.

In the current version of the data replacement algorithm, we entirely replace the most similar cases in the original dataset by the new data from the user. We also did a follow-up experiment to investigate the influence of an alternative replacement rule: we checked the dissimilarity value of the replaced cases and found that 80% of the replaced cases have a dissimilarity value less than 0.5. Based on this statistics, we recomputed the replacement with the most similar case replaced only if the dissimilarity value is less than 0.5 (threshold), in this way, we can keep 20% of the (relatively different) old data which otherwise would have been replaced also. The result shows, however, that the model accuracy is almost the same as before. Still we see the potential to find an optimal threshold to improve the model in the future.

In addition, we see several other opportunities to further extend our work, including

- the current retraining procedure is off-line, the ultimate goal is to make it real-time
- give users more flexibility to configure the brushing tool and even involve them to the brushing tool design
- explore other deep learning methods such as RNN and GAN to learn user's brushing behavior
- the extension of our principal approach to other views and the corresponding brushes

We hope that this work can inspire further related research, especially in visualization to offer the opportunity for users to improve the data-driven approach by their own knowledge and behavior.

REFERENCES

1. Jonathan C. Roberts. "State of the art: Coordinated & multiple views in exploratory visualization." *Proc. Coordinated and Multiple Views in Exploratory Visualization*, 2007, pp. 61-71.
2. Richard A. Becker and William S. Cleveland. "Brushing scatterplots." *Technometrics*, vol. 29, no. 2, 1987, pp. 127-142.
3. Helwig Hasuer. "Generalizing focus+ context visualization." *Scientific visualization: The visual extraction of knowledge from data*, 2006. pp. 305-327.
4. Allen R. Martin and Matthew O. Ward. "High dimensional brushing for interactive exploration of multivariate data." *Proc. the 6th Conference on Visualization*, 1995. p. 271.
5. Helmut Doleisch, Martin Gasser, and Helwig Hauser. "Interactive feature specification for focus+ context visualization of complex simulation data." *VisSym*. vol. 3, 2003, pp. 239-248.

6. Philipp, Muigg, et al. "A four‐level focus+ context approach to interactive visual analysis of temporal features in large scientific data." *Computer Graphics Forum*. vol. 27, no. 3, 2008, pp. 775-782.
7. Chaoran Fan, Helwig Hauser. "User-study based optimization of fast and accurate Mahalanobis brushing in scatterplots." *Proc. the 22nd International Symposium on Vision, Modeling and Visualization*, 2017, pp. 77-84.
8. Sanjin Radoš, et al. "Towards quantitative visual analytics with structured brushing and linked statistics." *Computer Graphics Forum*. vol. 35, no. 3, 2016, pp. 251-260.
9. Niklas Elmqvist, et al. "Fluid interaction for information visualization." *Information Visualization*, vol. 10, no. 4, 2011, pp. 327-340.
10. Stuart K. Card, George G. Robertson, and Jock D. Mackinlay. "The information visualizer, an information workspace." *Proc. the SIGCHI Conference on Human factors in computing systems*, 1991, pp. 181-186.
11. Chaoran Fan and Helwig Hasuer. "Fast and accurate CNN-based brushing in scatterplot." *Computer Graphics Forum*, vol. 37, no. 3, 2018, pp. 111-120.
12. Matěj Novotný, and Helwig Hauser. "Similarity brushing for exploring multidimensional relations." *Journal of WSCG*, 2006, vol.14, pp. 105-112.
13. Philipp Koytek, et al. "MyBrush: Brushing and Linking with Personal Agency." *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, 2017, pp. 605-615.
14. David H. Hubel and Torsten N. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." *The Journal of physiology*, vol. 160, no. 1, 1962, pp. 106-154.
15. Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition." *Competition and cooperation in neural nets*, 1982, pp. 267-285.
16. Yann LeCun, et al. "Gradient-based learning applied to document recognition." *Proc. IEEE*, vol. 86, no. 11, 1998, pp. 2278-2324.
17. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*, 2012, pp. 1097-1105.
18. Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning." *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, 2010, pp. 1345-1359.
19. Burr Settles. "Active learning." *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, 2012, pp. 1-114.
20. John W. Tukey and Paul A. Tukey. "Computer graphics and exploratory data analysis: An introduction." *The Collected Works of John W. Tukey: Graphics: 1965-1985* vol. 5, 1988, p. 419.

ABOUT THE AUTHORS

Chaoran Fan is a Ph.D candidate in the Department of Informatics, University of Bergen, Norway. His research interest is in information visualization, specifically for improving the user interaction in visual analytics by leveraging the machine learning knowledge. Contact him at Chaoran.Fan@uib.no

Helwig Hauser is a professor at the University of Bergen's Informatics Department, where he leads a visualization research group. His research interests are interactive visual analysis, illustrative visualization, the combination of scientific and information visualization and the application of visualization to various domains. Contact him at Helwig.Hauser@uib.no.