

Fast and accurate CNN-based brushing in scatterplots

Chaoran Fan and Helwig Hauser

University of Bergen, Norway, ii.UiB.no/vis

Abstract

Brushing plays a central role in most modern visual analytics solutions and effective and efficient techniques for data selection are key to establishing a successful human-computer dialogue. With this paper, we address the need for brushing techniques that are both fast, enabling a fluid interaction in visual data exploration and analysis, and also accurate, i.e., enabling the user to effectively select specific data subsets, even when their geometric delimitation is non-trivial. We present a new solution for a near-perfect sketch-based brushing technique, where we exploit a convolutional neural network (CNN) for estimating the intended data selection from a fast and simple click-and-drag interaction and from the data distribution in the visualization. Our key contributions include a drastically reduced error rate—now below 3%, i.e., less than half of the so far best accuracy—and an extension to a larger variety of selected data subsets, going beyond previous limitations due to linear estimation models.

1. Introduction

Linking and brushing is useful for interactive visual data exploration and analysis in coordinated multiple views [Mun14, Rob07]. Already 30 years ago, Becker and Cleveland [BC87] defined brushing as an interactive method for selecting data points in a visualization by drawing simple geometries onto it. A key functionality in coordinated multiple views is that brushing leads to a consistent highlighting of the selected data in all linked views. This results in the most common form of focus+context visualization [Hau05], enabling the fast and effective exploration of data relations, which are too challenging to show in just one view. Many techniques for brushing have been developed and 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 [KMG*06], etc.
- *lassoing*—the user selects subsets by drawing a geometrically detailed lasso around the target group of item representations
- *logical combinations of simple brushes*—the user makes use of multiple brushes and combines them using logical operators to refine the data selection [MW95, DGH03]
- *sketch-based brushing*—the user sketches a shape onto a visualization and a selection heuristic is used to determine which data are selected [MKO*08, FH17, RSM*16]

For designing a brushing technique, two particularly important criteria should be taken into account:

- *efficiency*—is the brushing is fast enough (including the interaction and all computation) to enable a fluid data exploration/analysis [EVMJ*11, TKBH17]?
- *accuracy*—does the brushing interaction lead to a selection of exactly the data subset, which the user wished to select in the

view (we refer to the most common form of brushing, where data points are selected due to their location in the visualization)?

Despite the rich 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 one point, 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 [CRM91]. 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 at the price of being slower—specifying a lasso, for example, easily becomes a unit task by itself [CRM91], potentially interrupting the exploration/analysis process. In our work, we aim to integrate both criteria in one technique as good as possible.

Recently, deep learning methods, especially convolutional neural networks (CNN), have been used very successfully in a wide range of fields including natural language processing [SHG*14, KGB14], object detection [RHGS15] and image classification [KSH12]. As brushing is mainly used to select spatially coherent data subsets, which is related to detecting patterns in images, we see the potential of exploiting deep learning to improve brushing even further.

Inspired by the impressive performance of CNNs in image processing, we developed a new CNN-based technique for brushing in scatterplots (we chose brushing in scatterplots as our study case, assuming that this approach is extensible to other views and according brushes, as well). Our quantitative evaluation shows that we reduce the brushing error rate from about 8% (Mahalanobis

brush [FH17]) to about 2.5%. We also build on a fast and simple click-and-drag interaction, but provide a solution which is much faster and also more flexible in terms of which data subsets can be selected (not limited due to a linear model). Since brushing is central in most modern visual analytics systems, we see this result as potentially very relevant.

This paper is organized as follows: After reviewing related work (Sect. 2), we first describe our principal approach (Sect. 3), before we then present our technique in detail (Sect. 4). The network training and evaluation are presented in sections 5 and 6, before we present details about our user studies (Sect. 7) and the model of natural variation among click-and-drag sketches for brushing (Sect. 8). We conclude and address future work in section 9.

2. 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 network.

2.1. Brushing techniques

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 is intrinsically based on the interaction between the user and the system, often a combination of mouse/cursor motions and clicks. Less usual methods, based on eye/head tracking, for example, or gestures in a virtual reality environment, have also been proposed [YA01].

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 [MW95], for example, enable the user to configure composite brushes by applying logical combinations of brushes, including unions, intersections, negations, and exclusive or operations.

Koytek et al. [KPV*17] 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. Hurter et al. [HTE11] developed a semantic lens which selects a specific spatial and attribute-related data range and it is applicable for scenarios requiring a mixed selection of the zones of interest.

Similarity brushing [NH06, MKO*08] 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 [RSM*16]. 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 then 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. Fan and Hauser [FH17] 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. 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.

2.2. CNNs and visualization

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 [HW62]. The concept of a neocognitron, proposed by Fukushima [FM82], is widely considered a fundamental basis of modern CNNs. LeCun et al. [LBBH98, LBD*90] 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. [KSH12] 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.

Considering an increasing number of successful applications of CNNs in many fields, we would expect according approaches also in visualization. While several interesting works look into the opportunity of visualization helping with the design, training, and analysis of CNNs [ZF14], we do not yet see a mentionable number of visualization solutions that exploit CNNs, in particular not in interaction techniques in visualization. With our work, we also hope to inspire interesting new research in this direction.

3. The principal approach

The overall goal of our research was to devise a brushing technique, which is both fast and accurate. In order to get as close as possible to both requirements, we used the following approach (also illustrated in Figure 1):

In order to achieve a fast interaction and a fluid exploration, we excluded any technique that would require the user to do multiple basic interactions in order to define just one brush (like a lasso, for example). To be as accurate as possible, we had to go beyond simple geometries with their limited abilities to accurately select data subsets, in particular in “crowded” regions of a visualization. Therefore, we needed a computational link between the fast and simple interaction and the selection of a non-trivially delimited subset, estimating the visual structure that the user identified as the

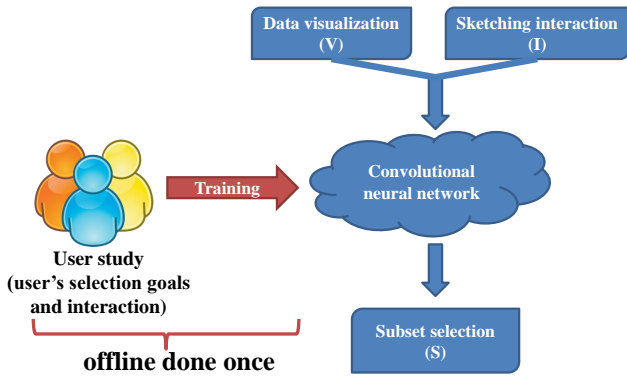


Figure 1: Illustration of our principal approach: To be fast, we use sketching as interaction; to estimate which data to actually brush, we use a CNN trained with data from two user studies.

brushing target. Usually, users brush subsets, which are spatially coherent in the visualization. Thus, we assume that we can estimate the brushing goal from both the actual brushing interaction and the data distribution in the visualization near the interaction.

Following successful previous work [MKO*08,RSM*16,FH17], we deemed the combination of a basic sketching interaction I , indicating the location, size, and orientation of the subset to brush, with a computational estimation function S , determining which subset to actually select, based on its visualization V near sketch I , to be a useful framework for modeling our solution. In previous work [MKO*08, RSM*16, FH17], the estimation function S was carefully modeled according to meaningful heuristics, based, for example, on a geometric similarity function in visualization space.

Since S amounts to interpreting the data visualization in terms of which spatially coherent subset best possibly relates to the sketching interaction, we found it promising to exploit recent successes of deep learning in image processing for our solution. We expected that the increased flexibility of this approach also helps to overcome limitations in previous work with respect to the variety of shapes that such an interaction can address—all solutions S for sketch-based brushing of scatterplots, so far [RSM*16, FH17], are limited to brushing structures that are described by linear models.

We also wished that the users would not have to adjust any off-screen parameters, interrupting their exploration/analysis (such that they can benefit from a fluid interaction with the data). Thus, we constructed our solution around a convolutional neural network (CNN) that we trained with data from two user studies.

The first user study, presented in more detail in another recent publication [FH17], provided information about both the brushing goals (which dataset subset did the users wish to brush) and the according interaction (which gesture would the user do to actually select the targeted data subset). In the second user study, presented further back in this paper, we examined the variation information of the user's interaction in order to use this for modeling an extension of the training data for the CNN.

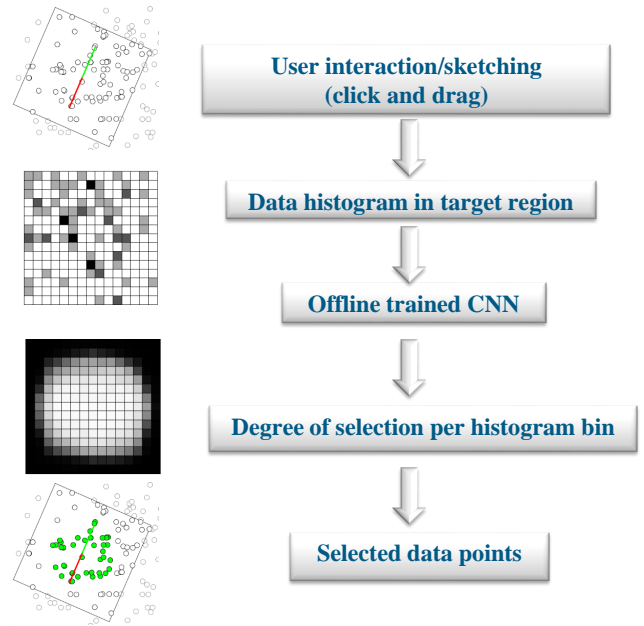


Figure 2: Overview of our fast and accurate brushing technique: For sketching, the user clicks into the middle of the data subset to be selected and drags the pointer to the border of the subset; The CNN then sees the data distribution near the interaction as a 2D histogram. It delivers a degree-of-selection value per histogram bin, from which we can compute, which data subset is selected.

4. The new brushing technique

Figure 2 provides an overview of our new brushing algorithm. In the following, we first describe the overall construction of our solution, before we then describe the individual components in detail.

4.1. Technique at large

Since we aim at estimating the selection information S from both the input sketch I as well as from the data visualization V , we need to efficiently and effectively consider these two heterogeneous parts of input information. For mainly two reasons, we handle I and V individually, using the CNN only for the interpretation of V . The critical input from the click-and-drag sketch, i.e., the click point c (center of the interaction) as well as the length r and the angle ϕ of the drag component, is first used to locate, scale, and orient the receptive field of the CNN. This way, we "normalize" the network's operation with respect to I by a simple linear transformation such that we can easily "undo" this normalization after the network's estimation process. Accordingly, the network's task is then to interpret the 2D data distribution in the appropriately located, scaled, and rotated region of the visualization. In order to predict which data subset to select, we model this step as an image processing operation: on the input side, we let the network see a 2D histogram of the data in the targeted area; on the output side we expect a measure p per bin of the histogram, indicating a "degree of selection" such that a simple thresholding at $\bar{p} = 0.5$ can identify the region within the target area corresponding to the selected data subset.

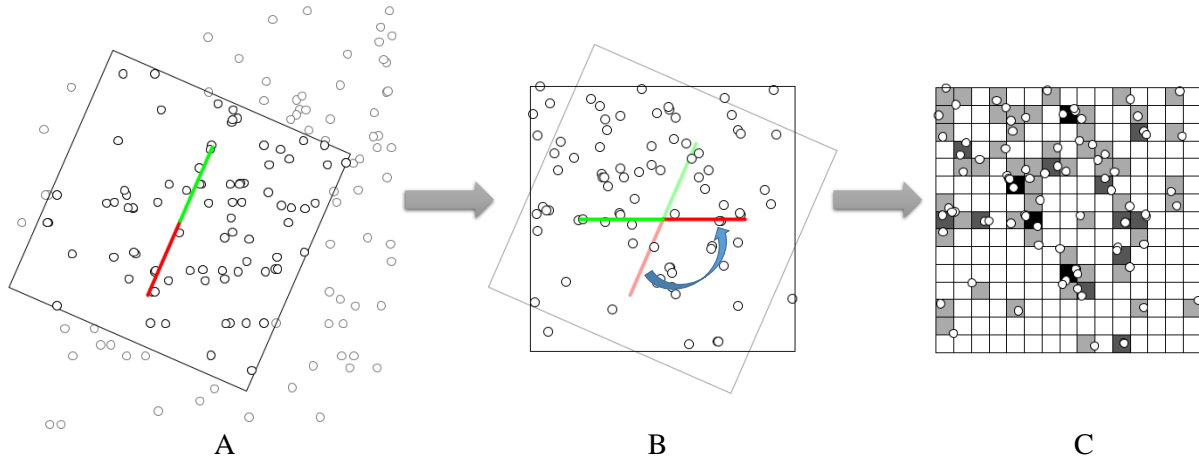


Figure 3: Computing the input to the CNN: A, a square area is specified by the interaction (red line segment); B, after rotating to the horizontal; C, histogram of the local data distribution (CNN input).

4.2. Computing the input to the CNN

The interaction I by the user amounts to a line in the scatterplot. To focus on the related, local visualization V near I , we use a square area including the user's brushing goal, with side length $2 \cdot r \cdot \omega$ and r being the length of I . To choose ω , we balance two goals: On the one hand, we need to make the receptive field of the network large enough to indeed see the data subset, which is targeted by the user as the brushing goal. On the other hand, this area should be not bigger than necessary in order to optimize the CNN results with respect to the resolution of the input histogram. Examining the user study data [FH17], we found $\omega = 1.5$ to be a useful compromise. Figure 3A shows the square area of the local visualization V related to I (red line segment).

In order to let the CNN see a normalized input (independent of I) as well as to interpret its output efficiently, the square area is rotated by $-\phi$ into a horizontal orientation as shown in Figure 3B. To make use of the local data distribution, we divide the square into a grid with a specific resolution (15 by 15 in our experiments) and compute a histogram by counting how many data points show up in each bin of the grid, denoted by C_{ij} where $i, j \in [1, 15]$. We then normalize the value of each bin into $[0, 1]$ by $C_{ij}/\max(C_{ij})$. Figure 3C shows a visualization of the network input with darker bin colors representing larger C_{ij} values.

4.3. CNN design

A typical image processing CNN is composed of convolutional, pooling and dense layers [LBD*90]. The purpose of convolutional layers is to extract patterns in local regions of the input images. Pooling layers are also referred to as a downsampling layers, with maxpooling being the most popular choice. This serves two main purposes. First, the number of parameters gets limited, reducing both the computation cost and overfitting. Second, the network can this way "see" on different scales, including also larger structures (earlier layers usually see smaller structures with subsequent layers then focussing on larger patterns). Fully connected layers then

connect every neuron in one layer to every neuron in the next layer. This is typically used in the last stages of the CNN.

For CNN design, the two most important goals are to avoid both overfitting and underfitting. Overfitting refers to when a model is overly tuned to the training data so that it does not generalize well. And if the model is too simple, with too few parameters, then this leads to underfitting, i.e., bad results (low accuracy, etc.).

We carefully experimented with many different layouts/settings of the CNN model, varying the size and number of the convolution filters, the number of convolutional and fully-connected layers, and the number of neurons in the fully-connected layers. As a result, we found a model which fits our scenario well. In our design, we deviate from the conventional CNN layout by replacing the last layer (classifier) with a structured regression layer to encode the output information from which the actual data subset selection can be derived in a subsequent step.

Altogether, we propose a model with two convolutional (C), two max-pooling (M), and two fully-connected layers (F). Figure 4 shows this architecture and the association between the last layer and the histogram-aligned grid of p -values. In detail, our model is configured as Input($15 \times 15 \times 1$), C($11 \times 11 \times 16$), M($6 \times 6 \times 16$), C($4 \times 4 \times 16$), M($2 \times 2 \times 16$), F(64), F(64), and F(225). The sizes of the C and M layers are defined as width \times height \times depth, where width \times height determines the extent of each feature map and depth represents the number of maps (filters).

The activation function of the last F layer (regression) is chosen to be a sigmoid function so that the output values are from $[0, 1]$. To reduce the likelihood of vanishing gradients, ReLU is used for all the other F and C layers. The filter size is chosen to be 5×5 for the first C layer and 3×3 for the second one. The max pooling layer uses a window of size 2×2 with a stride of 2 in each direction.

In order to check that we have chosen a reasonable number of parameters and a useful structure, avoid overfitting as well as underfitting, we have also visualized the weights of the neurons in

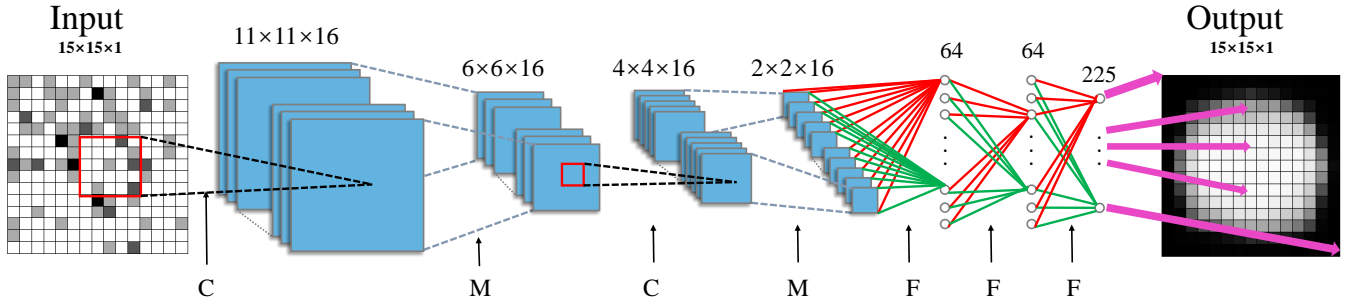


Figure 4: The proposed CNN model. *C*, *M* and *F* represent the convolutional layers, max-pooling layers, and fully connected layers, respectively. The purple arrows from the last layer illustrate the association between the final layer’s outputs and histogram-aligned grid of degree-of-selection values.

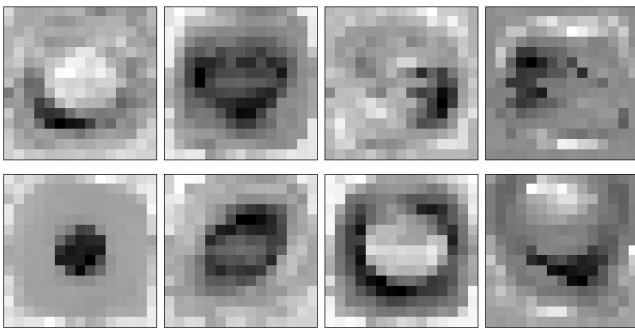


Figure 5: Visualization of the weights of 8 selected neurons among 64 in the second fully connected layer of the CNN.

the second fully connected layer, the output of which are used to compose the overall results. The weights are useful to visualize, because well-trained networks usually display nice and smooth filters without any noisy patterns [ZF14]. Noisy patterns can be an indicator of a network that has not been trained for long enough, or possibly a very low regularization strength that may have led to overfitting. Figure 5 shows a weights visualization of 8 selected neurons among 64 in the second fully connected layer, showing that our model learns meaningful structures and patterns.

4.4. Interpreting the output of the CNN

The output of the CNN is a grid comprised of degree-of-selection values p per histogram bin and we threshold this information at $\bar{p} = 0.5$ to locate the selected data subset (Fig. 6). We use the Marching Squares algorithm [LC87] with a threshold of 0.5 to generate the selection contour based on the two-dimensional network output and all points in the selection contour are selected to be the brushing result. Instead, one could also use the p values directly and select all data points that fall into a bin with $p > 0.5$. Since this would correspond to an unnatural selection contour, we prefer the smoother results as provided by the Marching Squares.

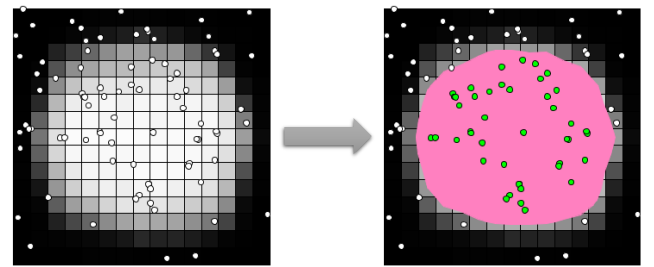


Figure 6: Left: output of the CNN. Right: the points colored in green are the brushing result (inside the Marching Squares contour, surrounding the pink area).

5. Training the CNN

We define the training data as $(x_i \in T_{in}, y_i \in T_{out})_{i=1}^N$ with pairs of input and expected output (N is the number of the training samples). We optimize the parameters of the network based on the training data using the mean-squared error as a loss function. The method of computing the network input x_i (appropriately located, scaled, and rotated histograms) has been described in section 4.2. In the following, we explain the design of the reference output y_i , which the model is trained against, and the implementation of the CNN.

5.1. Computing the reference output

For training the CNN, the information of the user goal (data items to select), which we have from the first user study, needs to be given to the training in an appropriate form that is compatible with the output layer of the CNN. For the specific square area that we consider for x_i , we know which points are the user’s goal from the user study. On the left side of Figure 7, for example, the yellow points are the user’s brushing goal and the red points are not to be selected.

To extract the information that is needed for the CNN training, we convert this binary select-vs.-disregard information from the user study into an image of the same resolution as the CNN input, using an adapted form of the K -nearest neighbors algorithm [Alt92]. For each bin of the grid, we estimate the degree-of-selection value p , that we then want the network to learn, by

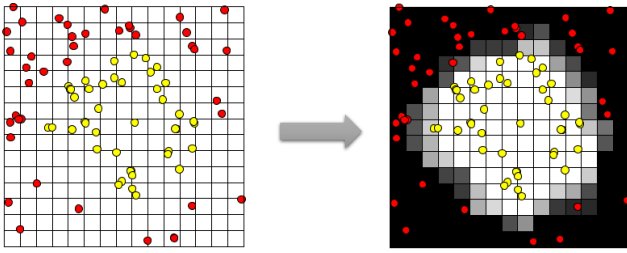


Figure 7: Left: binary select (yellow) vs. disregard (red) information from the user study. Right: the image-based reference output computed with an adapted nearest neighbor algorithm.

considering all data points in the bin and, if needed, nearby points. Further points are included from near to far, if there are less than K points in the center bin, stopping when at least K points are found. We give less weight to more distant points, based on the Euclidean distance d_k between the bin, where the point is located, and the center bin. We then estimate the degree-of-selection by

$$p = \frac{\sum_k i_k / (1 + d_k)}{\sum_k 1 / (1 + d_k)} \quad (1)$$

where $p \in [0, 1]$, k being the index of the points in the search area, and $i_k = 1$ if the point is a user goal, otherwise $i_k = 0$.

In general, the user goal is confined to the inside of the target area. Thus we ensure that the border bins have small values. Accordingly, we stop the search procedure, when an outside bin of the grid is searched. If the number of points found so far is then less than K , we synthesize the missing points right outside of the grid, and these points are labeled as not being a user goal.

A visualization of one according reference output is on the right of Figure 7: the darker a bin is colored, the smaller the corresponding p value is. Even though also other methods for estimating p come to mind, we found that this simple approach gave very good results—most likely due to well-behaved data from the user study (all of the user goals from the study lead to selection geometries with substantially smooth boundaries). In our research, we experimented with different values of K and validated them against the user study data—we found that $K = 3$ achieved the smoothest boundaries that successfully separate the user goal.

5.2. Training details

Usually, high accuracy cannot be achieved unless enough training samples are provided. It is labor-intensive and time-consuming to invite a large number of users to provide large amounts of user data. Instead, we follow a common strategy and synthesize additional training data [KSH12] from the already acquired training set based on a second user study that we did with the goal to study the variation of the user's interaction (in our sketch-based brushing context). We assumed that there would be a certain amount of natural variation in the users' sketching interaction (in terms of where exactly they click and how far and in which direction they drag). In the user study, we thus measured this variation, and modeled it in a statistically best-possible way, and then synthesized additional

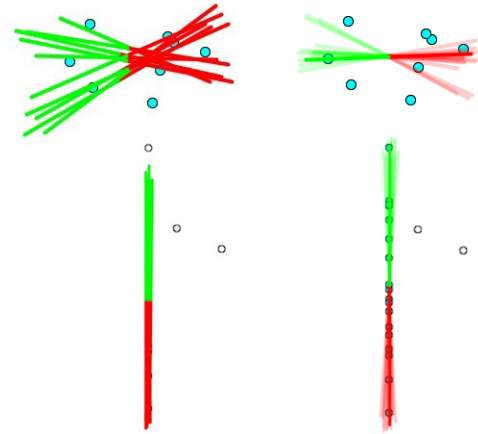


Figure 8: Left: Two cases from the second user study, 10 interactions for a specific brushing target in each case. Right: 15 modeled interactions per case (semi-transparent) with the original user interaction shown as solid line.

interaction sketches according to the resulting models as additional training data for the CNN. On the left of Figure 8, we see several user interactions from the second user study that we used for the variation analysis, and on the right are correspondingly modeled synthetic variations, confirming that our model leads to meaningful new training data. More details about this user study and the modeling procedure are provided in sections 7 and 8, respectively.

We then used three datasets of different sizes to train the CNN and compared their performances. The smallest dataset consists of 500 (x_i, y_i) -pairs, based on 500 selections from the first user study. The larger and the largest training sets were generated by synthesizing additional 1500 and 7500 (x_i, y_i) -pairs, respectively.

We implemented the network and executed its training in Keras [C*15] which provides useful GPU acceleration. For the training and testing, we used a PC with an Intel Xeon E5-1650 CPU and an NVIDIA GeForce GTX 1080 GPU. We used regularizers in the convolutional layers and a dropout function with a drop rate of 0.2 to avoid overfitting. As the output of our model is related to a degree-of-selection instead of a binary matrix, an L2 regularizer is more suitable in our model, resulting in less sparse output as when using an L1 regularizer. The learning rate was set to be 10^{-3} and in order to obtain a good convergence towards a high-quality optimum, we ran 10000 epochs for the training with a full batch.

6. Evaluation

For evaluating the new method, and in particular the trained CNN, we used k -fold cross-validation [K*95]. In k -fold cross-validation, the original sample is randomly partitioned into k equal sized sets. In each of the k folds, a single set is retained as the validation data for testing the model, and the remaining $k - 1$ sets are used as training data. The cross-validation process is then repeated k times, with each of the k sets used exactly once for validation. The results from all k folds are then averaged to assess the model. In our evaluation, we set $k = 6$ and split the original 600 selections (from the first user

Table 1: Statistics of the cross validation and training times

Group (validation)	Size factor of training data		
	1	4	16
G_1	95.73%	96.33%	96.65%
G_2	96.43%	97.74%	97.72%
G_3	98.39%	98.50%	98.60%
G_4	97.11%	97.13%	97.24%
G_5	95.00%	96.02%	96.82%
G_6	95.37%	97.17%	97.46%
<i>Mean</i>	96.34%	97.15%	97.42%
<i>Variance</i>	$1.3 \cdot 10^{-4}$	$6.9 \cdot 10^{-5}$	$4.1 \cdot 10^{-5}$
<i>Time</i>	≈ 5 mins	≈ 20 mins	≈ 80 mins

study) into six evenly sized groups G_i . For training the network, we use five groups as such (500 selections), or the extended training sets (see section 5.2) with 2000 or 8000 selections, respectively.

Table 1 shows the results of the cross validation in terms of accuracy for the three training set sizes. With the extended training data, the trained model is more stable and has a higher accuracy. We also see that the performance of our model, when using the 16 times larger training set, is only slightly better as when trained with the four times bigger training set, with an overall accuracy 97.42%.

Based on the trained CNN model, we did a quantitative accuracy comparison with the previously published new Mahalanobis brush [FH17] using the same 600 selections as presented in their user study. For each point in each of the cases, we test whether the Mahalanobis brush selects it, whether our new technique selects it, and whether it should be selected (ground truth), looking at 252,400 points altogether. We use different colors in the visualization to represent the comparison result:

- yellow points (both brushing techniques succeed to select the point correctly; both true positive), purple points (both brushing techniques fail to select, i.e., both false negative), and pink points (both techniques select falsely; both false positive)—purple and pink points (both techniques fail) amount to about 4.57% of all cases, where at least one technique fails.
- green points (the new technique succeeds, while the old fails) and blue points (the original method selected falsely, while the new one does not)—these points represent the cases, where our new technique improves the so far best results and $\approx 89.3\%$ of all cases, where at least one method fails, fall into this category!
- orange points (the new method fails to select, while the old did) and red points (the new method selects falsely, while the old did not)—these points represent cases, where the new technique is worse than the one (only about 6.13% of all cases, where at least one method fails, amount to this category).

To make this color-coding easier to follow, an accordingly colored Venn diagram is embedded in Figures 9 and 10 as color legend. The dashed circle on top represents the user goal (ground truth). The solid circle represents the brushing results by our new technique, while the dotted one surrounds the brushing results by the previously published Mahalanobis technique. We note that in the shown schematic, the areas do not correspond to the proportions of the respective cases—it’s just a color legend.

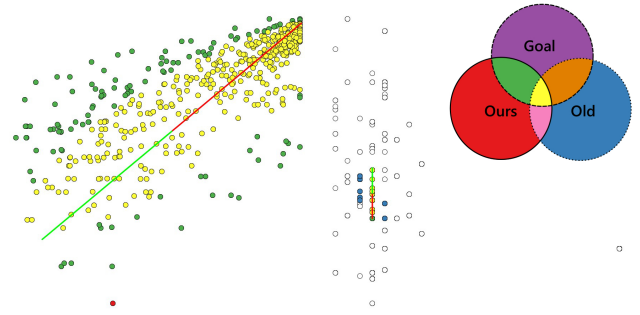


Figure 9: Two typical examples of a good match between the user’s goal and the CNN-based brushing technique.

In total, when using the dice coefficient [Dic45] to assess how well both techniques agree with the ground truth, we get excellent 99% for our new technique, as compared to 91% for the reference method [FH17]. In terms of efficiency, the new technique is similarly fast as the previous Mahalanobis brush for small subsets; when brushing 2000 points, for example, it takes around 20ms. But when it comes to larger datasets, our method takes only 180ms when brushing 1 million points, while the Mahalanobis brush takes very long 110s for 100000 points, which is orders of magnitude too slow for a fluid interaction with large data.

To further substantiate the evaluation of our new approach, we organized a new user study and invited ten users to test our CNN model on new data. In this study, we followed the established procedure of the first user study [FH17], but provided 6 completely new datasets (D7–D12 in the supplementary video) for the users to brush, which were not used in any way in the construction or training of our model. We got 120 new selections from this user study and the average accuracy is $\approx 95.3\%$, providing further evidence that our model is good at capturing relevant features of the user’s brushing preference, rather than being biased by the training data.

6.1. Examples of good cases

Figure 9 shows two typical situations, when our method performs very well, while the Mahalanobis brush [FH17] results in a worse selection. On the left of Figure 9 we see many green points which the Mahalanobis brush would need to select, but actually does not, while our technique selects them correctly. Besides, we see many blue points on the right of Figure 9: by design, the Mahalanobis brush brushes an elliptical area that more often than the CNN has troubles in selecting elongated, skinny groups.

6.2. Worst cases

We also did a worst case analysis, shown in Figure 10, which we selected based on the ratio of false negatives to our brushing result (FN/ours) and false positives to the goal (FP/goal). Comparably large values in either of these measures identify our worst cases.

Case A, highest value of FN/ours: our technique has a problem to differentiate whether the user’s goal is a circular region or an elongated group in some very similar regions.

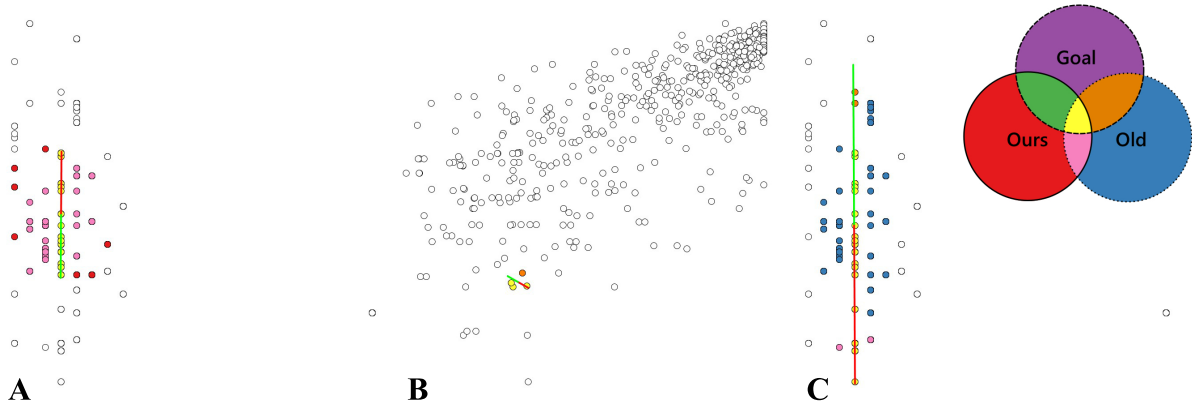


Figure 10: Three (most extreme) cases of suboptimal matches between the user’s goal and the new brushing technique.

Case B, highest value of FP/goal: the user’s interaction has a big influence when selecting very small subsets (here, the start point of the user interaction deviates a bit from the center point of the target cluster, leading to a bad performance in this case).

Case C: Actually, this is not a very bad example, but we chose to show it here, because it performs relatively badly both in FP/goal and FN/ours (a worst case that isn’t really bad, after all).

7. User studies

As the user plays a central role in brushing, we conducted a new user study to explore how the user uses our brushing tool in practice and analyzed the natural variation of the sketching interaction in order to prepare the synthesis of additional data for the training of the CNN. This new user study can be seen as a follow-up user study, based on the user study done earlier [FH17] (called the first user study in this paper). In the following, we first describe, what data we used from the earlier published user study, before then going into details about our new user study.

The previously published user study (50 participants) provided the following data which we also used for this work: Given a particular scatterplot (one out of six) and a particular request (one out of three), the study participant chose a target data subset to select (ground truth, reported by the participants using a lasso tool) and then also provided the corresponding click-and-drag interaction, which this participant would use to select the target group.

Naturally, for each user goal, the interaction done by the user will be at least a bit different every time. In order to understand the natural variation of the user’s sketching interaction, when having the same brushing operation in mind, and to model this variation to enable the synthesis of additional data for the training of the CNN, we did this follow-up user study based on the first user study. In the new study, 10 individuals, all students or employees from the Delft University of Technology, participated. We used 100 representative selections from the 600 selections in the first user study. For each user, 10 different selections were displayed only showing the user goal information.

The second user study then consisted of two parts: In the first

part, the users were asked to look at the points, which they should brush, using the user goal information from the first study for each case. Then, the users were required to use our new technique to select the target points. To do so, the users had to click into the center of the target points and then drag the pointer, while holding down the button of the mouse, to the border of the target group (and then release to finish the selection). The users were asked to repeat this interaction 10 times in each case. The user interaction (the start point and end point) was recorded. This resulted in 1000 interactions which we then studied in a variation analysis.

Before the start of the study, we presented our new brushing in a training session, where we showed the new technique by brushing a test dataset. These sessions took approximately 10 minutes and the participants were free to interrupt for questions and to take over the software to experiment with the new brushing technique until they were comfortable to do the study.

As mentioned, we also performed a third study to collect additional evidence about whether the learned model would generalize to new data and new users, basically following the design of the first one [FH17]. We provided six completely new datasets and invited ten new users. In the supplementary material, we include more details about this third study and detailed information about the achieved accuracy.

8. Modeling the variation in sketching

In order to make the CNN training as successful as possible, we extended the training data with synthetic interaction data based on the second user study. In our observation, the variation of the user interaction consists of three parts that meaningfully are modeled separately: the start point of the interaction (denoted by c), as well as the length r and the angle ϕ of the drag-component.

If a user brushes a big group, the potential variation of the length of the drag-component will be larger than when brushing a small group. Therefore, when considering the variation of interaction length r , we normalize by the mean of the interaction length. Further, the users’ interaction has much less angle variation when they brush an elongated, anisotropic group, compared with brushing

more isotropic subsets. Our user study cases are related to a specific question, which was used to instruct the users before brushing. The questions "select a small cluster" (S) and "select a big cluster" (B) are used to expect a rather isotropic data subset from the user and the anisotropic cases are more related to the "select an elongated group" (E) question. Therefore, we compute the variation of angles separately according to the two different types of cases. We used the statistical tool EasyFit [Sch02] to analyze which distribution fits our variation data best and the resulting function details (probability density functions) and their specific parameters are listed in Table 2.

For synthesizing new training data, given a user interaction I with $c = (c_x, c_y)$, r , and ϕ , we compute a new user interaction I' based on random samples from the accordingly fitted PDFs. The entries in the first column of Table 2 describe the random samples which we are drawing from the corresponding PDFs (with "var" referring to variance). The new I' is then given by c' , r' , and ϕ' according to $r' = r \pm \sqrt{\frac{var(r)}{mean(r)}} \cdot r$ and $\phi' = \phi \pm \sqrt{var(\phi)}$. To compute c' , the sampling result $var(c)$ is considered to be an intermediate variation as the variation of the start point c is also related to the size of the brushing goal. Therefore, we normalized for this relation before fitting the PDFs. As a result, we need to "undo" this normalization after sampling the PDF and get $c' = c \pm (var(c) + 0.1) \cdot (v + 20)/10$, where v is the standard deviation of the user goal (in x or y direction, respectively).

9. Conclusion and future work

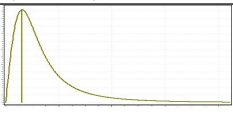
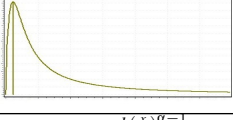
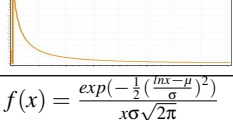
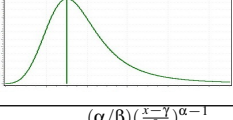
With this paper, we demonstrate how deep learning can be used to further improve the central operation of brushing in visual analytics. By learning the relation between the data subset to be selected and a click-and-drag sketch by the user to do the selection, we achieve a solution, which is both very fast and also very accurate. To the best of our knowledge, this is the first study to report the successful application of a structured regression model, realized by a convolutional neural network, to improve a central user interaction in visual analytics—in our case brushing in scatterplots. We demonstrate, quantitatively, and in comparison with the previously published Mahalanobis brush, that our CNN-based solution leads to a significant reduction of the error rate (from $\approx 8\%$ to $\approx 2.5\%$), while enabling very fast interaction. In the future, we see several opportunities to further extend our work, including

- the design of a brushing tool which is tailored for a single user, using an appropriate method to learn a user's particular brushing behavior over time
- research possible improvements of both the network input (using a different method for capturing the data distribution, like KDE) and the reference output (alternative ways to estimate p from the binary user goal information)
- the extension of our principal approach to other views and according brushes

Acknowledgements

We thank the participants of the user studies and TU Delft for the hardware support (a fast graphics card). We are also grateful to E. Gröller and A. Lundervold for fruitful discussions about

Table 2: Best-fit modeling of the observed variation in the executed sketches, when intending to brush the same data subset

Variance	Cases	Best-fit Probability Density Function	
$\frac{var(r)}{mean(r)}$	all	Burr	
		mode=0.30682	
		$k = 1.0164$	
		$\alpha = 1.9863$	
		$\beta = 0.53886$	$f(x) = \frac{\alpha k (\frac{x}{\beta})^{\alpha-1}}{\beta (1 + (\frac{x}{\beta})^\alpha)^{k+1}}$
$var(\phi)$	SUB	Burr	
		mode=0.00112	
		$k = 0.31523$	
		$\alpha = 1.6348$	
		$\beta = 0.00191$	$f(x) = \frac{\alpha k (\frac{x}{\beta})^{\alpha-1}}{\beta (1 + (\frac{x}{\beta})^\alpha)^{k+1}}$
$var(\phi)$	E	Lognormal	
		mode=3.211E-5	
		$\sigma = 1.8456$	
		$\mu = -6.9401$	
			$f(x) = \frac{exp(-\frac{1}{2}(\frac{\ln x - \mu}{\sigma})^2)}{x\sigma\sqrt{2\pi}}$
$var(c)$	all	Log-logistic	
		mode=-0.0241	
		$\alpha = 3.7167$	
		$\beta = 0.08711$	
		$\gamma = -0.09919$	
			$f(x) = \frac{(\alpha/\beta)(\frac{x-\gamma}{\beta})^{\alpha-1}}{(1+(\frac{x-\gamma}{\beta})^\alpha)^2}$

the CNN design. In addition, we highly appreciate the help of N. Pezzotti and A. Vilanova with optimizing the CNN architecture.

References

[Alt92] ALTMAN N. S.: An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185. 5

[BC87] BECKER R. A., CLEVELAND W. S.: Brushing scatterplots. *Technometrics* 29, 2 (1987), 127–142. 1

[C*15] CHOLLET F., ET AL.: Keras, 2015. 6

[CRM91] CARD S. K., ROBERTSON G. G., MACKINLAY J. D.: The information visualizer, an information workspace. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems* (1991), CHI '91, ACM, pp. 181–186. 1

[DGH03] DOLEISCH H., GASSER M., HAUSER H.: Interactive feature specification for focus+context visualization of complex simulation data. In *Proc. of the Symp. on Data Visualisation* (2003), VISSYM '03, pp. 239–248. 1

[Dic45] DICE L. R.: Measures of the amount of ecologic association between species. *Ecology* 26, 3 (1945), 297–302. 7

[EVMJ*11] ELMQVIST N., VANDE MOERE A., JETTER H.-C., CERNEA D., REITERER H., JANKUN-KELLY T. J.: Fluid interaction for information visualization. *Information Visualization* 10, 4 (Oct. 2011), 327–340. 1

[FH17] FAN C., HAUSER H.: User-study based optimization of fast and accurate Mahalanobis brushing in scatterplots. In *Vision, Modeling and Visualization* (2017), The Eurographics Association. doi:10.2312/vmv.20171262. 1, 2, 3, 4, 7, 8

[FM82] FUKUSHIMA K., MIYAKE S.: Neocognitron: A self-organizing

- neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285. 2
- [Hau05] HAUSER H.: Generalizing focus+context visualization. In *Scientific Visualization: The Visual Extraction of Knowledge from Data*, Bonneau, Ertl, Nielson, (Eds.). Springer, 2005, pp. 305–327. 1
- [HTE11] HURTER C., TELEA A., ERSOY O.: Moleview: An attribute and structure-based semantic lens for large element-based plots. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2600–2609. 2
- [HW62] HUBEL D. H., WIESEL T. N.: Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology* 160, 1 (1962), 106–154. 2
- [K*95] KOHAVI R., ET AL.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In *the 14th international joint conference on Artificial intelligence* (1995), vol. 14, Stanford, CA, pp. 1137–1145. 6
- [KGB14] KALCHBRENNER N., GREFFENSTETTE E., BLUNSOM P.: A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* (2014). 1
- [KMG*06] KONYHA Z., MATKOVIĆ K., GRAČANIN D., JELOVIĆ M., HAUSER H.: Interactive visual analysis of families of function graphs. *IEEE Transactions on Visualization and Computer Graphics* 12, 6 (Nov 2006), 1373–1385. 1
- [KPV*17] KOYTEK P., PERIN C., VERMEULEN J., ANDRÉ E., CARPENDALE S.: Mybrush: Brushing and linking with personal agency. *IEEE Transactions on Visualization and Computer Graphics* (2017). 2
- [KSH12] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105. 1, 2, 6
- [LBBH98] LECUN Y., BOTTOU L., BENGIO Y., HAFFNER P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324. 2
- [LBD*90] LECUN Y., BOSER B. E., DENKER J. S., HENDERSON D., HOWARD R. E., HUBBARD W. E., JACKEL L. D.: Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems* (1990), pp. 396–404. 2, 4
- [LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. In *ACM SIGGRAPH Computer Graphics* (1987), vol. 21, ACM, pp. 163–169. 5
- [MKO*08] MUIGG P., KEHRER J., OELTZE S., PIRINGER H., DOLEISCH H., PREIM B., HAUSER H.: A four-level focus+context approach to interactive visual analysis of temporal features in large scientific data. *Computer Graphics Forum* 27, 3 (2008), 775–782. 1, 2, 3
- [Mun14] MUNZNER T.: *Visualization Analysis & Design*. CRC Press, 2014. 1
- [MW95] MARTIN A. R., WARD M. O.: High dimensional brushing for interactive exploration of multivariate data. In *Proc. of the 6th Conf. on Visualization* (1995), VIS '95, IEEE Computer Society, pp. 271–278. 1, 2
- [NH06] NOVOTNÝ M., HAUSER H.: Similarity brushing for exploring multidimensional relations. In *Journal of WSCG* (2006), vol. 14, Václav Skala-UNION Agency. 2
- [RHGS15] REN S., HE K., GIRSHICK R., SUN J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (2015), pp. 91–99. 1
- [Rob07] ROBERTS J.: State of the art: Coordinated multiple views in exploratory visualization. In *Proc. of CMV '07* (July 2007), pp. 61–71. 1
- [RSM*16] RADOŠ S., SPLECHTNA R., MATKOVIĆ K., ĐURAS M., GRÖLLER E., HAUSER H.: Towards quantitative visual analytics with structured brushing and linked statistics. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 251–260. 1, 2, 3
- [Sch02] SCHITTKOWSKI K.: Easy-fit: a software system for data fitting in dynamical systems. *Structural and Multidisciplinary Optimization* 23, 2 (2002), 153–169. 9
- [SHG*14] SHEN Y., HE X., GAO J., DENG L., MESNIL G.: Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web* (2014), ACM, pp. 373–374. 1
- [TKBH17] TURKAY C., KAYA E., BALCISOY S., HAUSER H.: Designing progressive and interactive analytics processes for high-dimensional data analysis. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan 2017), 131–140. 1
- [YA01] YANG M.-H., AHUJA N.: *Face detection and gesture recognition for human-computer interaction*, vol. 1. Springer Science & Business Media, 2001. 2
- [ZF14] ZEILER M. D., FERGUS R.: Visualizing and understanding convolutional networks. In *European conference on computer vision* (2014), Springer, pp. 818–833. 2, 5