Online Appendix to: Simulation-based Optimization of User Interfaces for Quality-assuring Machine Learning Model Predictions

YU ZHANG, University of Oxford
MARTIJN TENNEKES and TIM DE JONG, Statistics Netherlands
LYANA CURIER, Open University of the Netherlands
BOB COECKE and MIN CHEN, University of Oxford

A ESTIMATING OPERATOR TIME COSTS

A.1 Experiment

We aim to estimate t_{new}, t_{view}, and t_{single} as outlined in Section 5.3 in the main text. To facilitate the estimation, we need to gather observations of $< T_{round}, N_{new}, N_{view}, N_{single} >$ through the experiment. We vary N_{new}, N_{view}, N_{single} in experiment trials and measure time cost T_{round}. N_{new} is the number of times the user activate the "new batch" command to request a new batch of data objects to quality-assure. N_{view} is the number of times the user carries out the "viewing" action to comprehend the visual representation of a single data object. N_{single} is the number of times the user activates the "single edit" command to change the label of a data object.

A.1.1 Apparatus. The experiment is conducted on a laptop with a 15-inch 3:2 screen. The screen resolution is 2560 × 1600. The screen is viewed by the participant from approximately 80 cm away. The participant used a physical mouse to conduct the tasks. The mouse sensitivity is 2000 DPI.

The experiment is carried out for three quality assurance interfaces as shown in Figures 2(a)–(c) in the main text. The QA4ML interfaces are all implemented as web applications that run in a browser. We implemented a browser plugin to log observations of T_{round}. One interface is for data extraction from visualization images that require quality assurance for JS-block and JS-grouping (see Figure 1(a) in the main text). The other interface is for solar panel detection from remote sensing images that requires quality assurance for SP-image (see Figure 1(b) in the main text). Each quality assurance interface presents a grid panel with each grid cell corresponding to a data object.

A.1.2 Procedure. The experiment is carried out through self-experimentation by the author for proof of concept. As the developer of the interfaces, the user is familiar with all the interface functionalities. The user has much experience using the interfaces before the experiment and represents the user group with high expertise in using the interfaces.

We carried out in total 332 trials, as shown in Tables 1 and 2 for the three different applications, with different grid layouts and different numbers of grid cells shown in the interfaces.

Before starting each trial, the browser plugin adds a white overlay to the interface and highlights the location of the grid cells. After the user clicks the overlay, the timer of the trial starts. The user

2160-6455/2024/01-ART1 $15.00
https://doi.org/10.1145/3594552

quality-assures the labels with the provided functionality of the interface. Once the user clicks the confirm button, the timer stops, and the time cost of the trial is logged. Each trial is conducted

Table 1. Experiment Design: The Number of Repeated Trials Conducted for Each Combination of the Independent Variables: Application, Grid Layout, Number of Data Objects

Application	Layout	#Objects	#Trials
JS-block	1 × 1	0	1
JS-block	1 × 1	1	20
JS-block	1 × 2	0	1
JS-block	1 × 2	1	15
JS-block	1 × 2	2	15
JS-block	2 × 2	0	1
JS-block	2 × 2	2	6
JS-block	2 × 2	4	15
JS-block	2 × 3	0	1
JS-block	2 × 3	3	10
JS-block	2 × 3	6	10
JS-block	3 × 4	0	1
JS-block	3 × 4	4	10
JS-block	3 × 4	8	10
JS-block	3 × 4	12	10
JS-block	4 × 5	0	1
JS-block	4 × 5	10	4
JS-block	4 × 5	15	6
JS-block	4 × 5	20	6
JS-block	6 × 6	0	1
JS-block	6 × 6	12	4
JS-block	6 × 6	18	3
JS-block	6 × 6	24	5
JS-block	6 × 6	30	6
JS-block	6 × 6	36	7
JS-block	8 × 8	0	1
JS-block	8 × 8	32	4
JS-block	8 × 8	48	5
JS-block	8 × 8	64	3

twice to alleviate the impact of the learning effect on the stability of time costs. The time is logged when the trial is conducted the second time. To simulate different levels of default label accuracies, we programmatically randomize the default labels of data objects. For example, for the binary classification case, given an accuracy *acc*, the program randomly selects $round(acc \cdot n_{batch})$ data objects to flip their true labels.

A.2 Initial Estimations of Operator Time Costs

Using the experiment data, we estimate t_{new}, t_{view}, t_{single} with multiple linear regression. Specifically, we go through the following estimation procedure for each combination of <application, layout>. We fit an individual model for each combination of <application, layout> because the application and layout may significantly influence the operation time costs.

Given <application, layout>, assume T_{round} follows a multiple linear model with regards to N_{new}, N_{view}, N_{single} as $T_{round} = N_{new}t_{new} + N_{view}t_{view} + N_{single}t_{single}$ in Equation (1) in the

Table 2. Experiment Design Continued: The Number of Repeated Trials Conducted for Each Combination of the Independent Variables: Application, Grid Layout, Number of Data Objects

Application	Layout	#Objects	#Trials
JS-grouping	1 × 1	0	1
JS-grouping	1 × 1	1	20
JS-grouping	2 × 3	0	1
JS-grouping	2 × 3	3	7
JS-grouping	2 × 3	6	6
JS-grouping	3 × 4	0	1
JS-grouping	3 × 4	4	3
JS-grouping	3 × 4	8	3
JS-grouping	3 × 4	12	3
JS-grouping	4 × 5	0	1
JS-grouping	4 × 5	10	5
JS-grouping	4 × 5	15	5
JS-grouping	4 × 5	20	5
JS-grouping	6 × 6	0	1
JS-grouping	6 × 6	12	5
JS-grouping	6 × 6	24	5
JS-grouping	6 × 6	36	5
JS-grouping	8 × 8	0	1
JS-grouping	8 × 8	24	1
JS-grouping	8 × 8	32	4
JS-grouping	8 × 8	48	5
JS-grouping	8 × 8	64	3
SP-image	1 × 1	0	1
SP-image	1 × 1	1	5
SP-image	2 × 3	0	1
SP-image	2 × 3	3	4
SP-image	2 × 3	6	4
SP-image	3 × 4	0	1
SP-image	3 × 4	4	4
SP-image	3 × 4	8	4
SP-image	3 × 4	12	4
SP-image	4 × 5	0	1
SP-image	4 × 5	10	4
SP-image	4 × 5	15	4
SP-image	4 × 5	20	4
SP-image	5 × 8	0	1
SP-image	5 × 8	16	4
SP-image	5 × 8	24	4
SP-image	5 × 8	32	4
SP-image	5 × 8	40	4

Table 3. Initial Estimation of Operator Time Costs: The Estimated Time Cost of Unit User Operations using Multiple Linear Regression

Application	Layout	t_{new} (ms)	t_{view} (ms)	t_{single} (ms)	R^2
JS-block	1 × 1	463 (± 5) **	518 (± 35) **	16 (± 47)	0.998
JS-block	1 × 2	441 (± 8) **	352 (± 33) **	148 (± 49) **	0.993
JS-block	2 × 2	778 (± 10) **	115 (± 25) **	247 (± 46) **	0.997
JS-block	2 × 3	773 (± 7) **	78 (± 12) **	291 (± 24) **	0.999
JS-block	3 × 4	923 (± 18) **	92 (± 16) **	206 (± 28) **	0.993
JS-block	4 × 5	878 (± 23) **	196 (± 15) **	226 (± 28) **	0.997
JS-block	6 × 6	873 (± 39) **	239 (± 12) **	215 (± 21) **	0.995
JS-block	8 × 8	946 (± 68) **	266 (± 18) **	322 (± 38) **	0.997
JS-grouping	1 × 1	461 (± 7) **	515 (± 40) **	66 (± 59)	0.997
JS-grouping	2 × 3	824 (± 12) **	191 (± 25) **	258 (± 45) **	0.998
JS-grouping	3 × 4	782 (± 10) **	190 (± 19) **	245 (± 35) **	0.999
JS-grouping	4 × 5	827 (± 17) **	199 (± 14) **	215 (± 25) **	0.998
JS-grouping	6 × 6	937 (± 22) **	252 (± 10) **	162 (± 18) **	0.998
JS-grouping	8 × 8	981 (± 46) **	292 (± 14) **	221 (± 25) **	0.998
SP-image	1 × 1	416 (± 9) **	999 (± 103) **	566 (± 145) *	0.999
SP-image	2 × 3	757 (± 20) **	155 (± 151)	701 (± 234) *	0.997
SP-image	3 × 4	861 (± 78) **	660 (± 198) **	285 (± 423)	0.976
SP-image	4 × 5	928 (± 84) **	506 (± 142) **	513 (± 295)	0.988
SP-image	5 × 8	912 (± 214) **	577 (± 252) *	473 (± 479)	0.976

For each estimation, the estimated standard error is shown in the parenthesis. The statistical significance is marked. ** means $p \leqslant 0.01$ and * means $p \leqslant 0.05$. R^2 columns show the R-squared of the entire linear model $T_{round} = N_{new}t_{new} + N_{view}t_{view} + N_{single}t_{single}$. It can be seen that for all the experimented applications and grid layouts, N_{new} contributes linearly to T_{round}. In most cases, N_{view} and N_{single} contributes linearly to T_{round}.

main text. We fit a multiple linear regression model for T_{round} to solve for t_{new}, t_{view} and t_{single}. The standard error, R^2, and significance of the fitting are computed. Table 3 and Figure 1 show the estimation result.

B MODELING AND REESTIMATING OPERATOR TIME COSTS

In the following, we introduce a process of smoothing the operator time costs estimated in Section A.2. We fit curves for the operator time costs and use the values on the curves to substitute the initial estimations of the operator time costs in Section A.2.

B.1 Model t_{new}

Goal. Reestimate t_{new} as a function of interface parameter n_{batch} to smooth the estimation.

Procedure. There are 19 combinations of <application, layout> and thus 19 $< n_{batch}, t_{new} >$ samples. We fit a model for all three applications combined. We have used the Nonlinear Regression Tool [3] to produce a set of candidate model functions $t_{new} = f(n_{batch})$.

- With the number of parameters = 2, the top 3 functions are
  - $y = \frac{ax}{x+b}$ with $R^2 = 0.9205$
  - $y = ae^{b/x}$ with $R^2 = 0.8989$
  - $y = a + b/x$ with $R^2 = 0.8630$

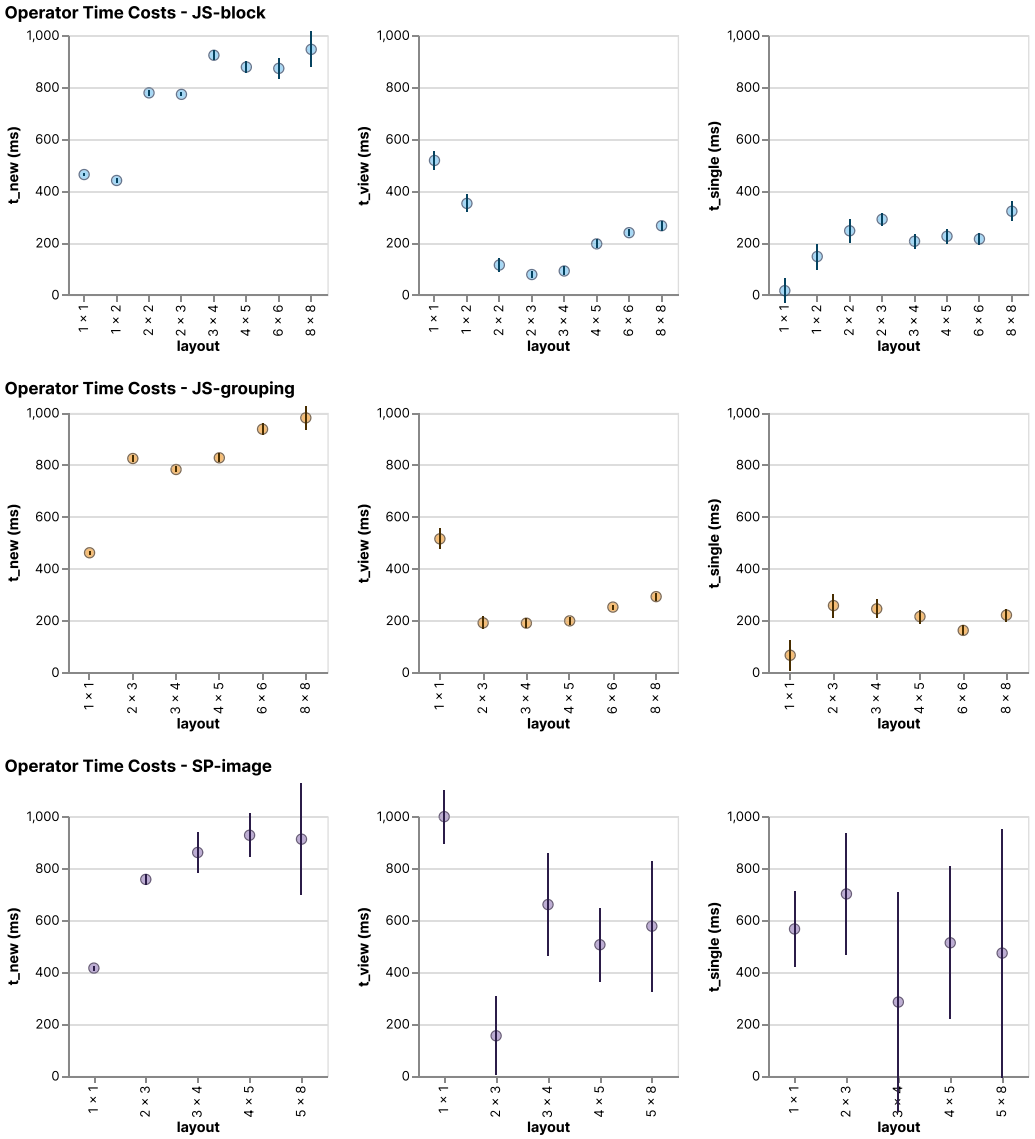

Fig. 1. Initial estimations of operator time costs: The estimated operator time costs t_{new}, t_{view}, and t_{single} in relation to the grid layouts for all the applications. The error bars show the estimated standard error of the estimation. As the number of grid cells grows, t_{new} increases, t_{view} first increases then decreases, t_{single} first increases then stays stable.

- With the number of parameters = 3, the top 3 functions are
  $-y = a + b/x + c/x^2$ with $R^2 = 0.9381$
  $-y = ax + \frac{bx}{x+c}$ with $R^2 = 0.9231$
  $-y = \frac{1}{ax^b+c}$ with $R^2 = 0.9225$
- With the number of parameters = 4, the top 3 functions are
  $-y = a + b/x + c/x^2 + d/x^3$ with $R^2 = 0.9486$
  $-y = a + b/x + c/x^2 + dx$ with $R^2 = 0.9486$
  $-y = a + b/x + c/x^2$ with $R^2 = 0.9381$

It can be seen that $y = a + b/x + c/x^2$ is reoccurring. It is the best when #parameters $= 3$ and the third best when #parameters $= 4$. Adding additional terms generate the best function when #parameters $= 4$. Removing the c/x^2 term generates the third-best when #parameters $= 2$.

We refer to the literature for more function options. The "new batch" button clicking relates to the object-pointing task in Fitts' law [1]. Fitts' law suggests that the object-pointing time cost can be modeled as $T = a + blog(\frac{A}{W} + 1)$. W is the size of the target object, A is the distance from the initial cursor position to the target object, whereas a and b are parameters to be fitted, which may depend on interface and apparatus settings.

To activate the "new batch" command, the user needs to move the cursor to the "new batch" button and click it. This procedure is similar to the Fitts' law setting, while a significant difference from typical Fitts' law settings is that the distance to the target (the "new batch" button) is not well defined. Because the last position of the cursor before the user decides to activate the "new batch" command can be an arbitrary point in the interface.

We adopt the following approximation to examine whether Fitts' law works for our scenario. Assume the aspect ratio of the grid cells does not change with the increase of n_{batch}. Denote the layout as $xf \times yf$ where x, y, f are integers. The maximal distance of points in the interface is thus $\sqrt{x^2 + y^2}f$. As f increases, the number of grid cells xyf^2 increases at the rate of f^2, and the maximal distance $\sqrt{x^2 + y^2}f$ increases at the rate of f. We approximate the distance A in Fitts' law by the maximal distance $\sqrt{x^2 + y^2}f$ and then represent it as $c\sqrt{n_{batch}}$ where c is an interface related constant. Thus, we get $t_{new} = a + blog_2(c\sqrt{n_{batch}} + 1)$. However, this function is not convex and is hard to optimize. Therefore, we modify it to $t_{new} = a + blog_2(\sqrt{n_{batch}} + 1)$ by removing c.

In short, we additionally consider the model function $t_{new} = a + blog_2(\sqrt{n_{batch}} + 1)$ to see whether Fitts' law applies to our scenario.

Thus, we choose $y = a + b/x + c/x^2$ and $y = a + blog_2(\sqrt{x} + 1)$ to be the candidate function families for t_{new}. We fit the models with linear regression.

Outcome.

- For $y = a + b/x + c/x^2$, the function fitted is:

$$t_{new} = 958.0085 - \frac{1254.8737}{n_{batch}} + \frac{739.4750}{n_{batch}^2}$$
$$\text{with adjusted } R^2 = 0.9303, SE = 49.4461$$

- For $y = a + blog(\sqrt{x} + 1)$, the function fitted is:

$$t_{new} = 280.0475 + 341.5430 \cdot log_2(\sqrt{n_{batch}} + 1)$$
$$\text{with adjusted } R^2 = 0.8004, SE = 83.6951$$

The fitting of $y = a + blog(\sqrt{x} + 1)$ is worse than $y = a + b/x + c/x^2$ in terms of adjusted R^2 and SE. It implies that the model suggested by Fitts' law is not numerically accurate in this scenario. Thus, we decide to adopt the model in Figure 2:

$$\mathbf{t_{new} = 958.0085 - \frac{1254.8737}{n_{batch}} + \frac{739.4750}{n_{batch}^2}} \quad (1)$$

B.2 Reestimate t_{view} and t_{single} by t_{new} Model

Goal. Reestimate t_{view} and t_{single} by removing the previous modeled t_{new} from the equations to possibly reduce the noise and make sure the original equations still approximately hold.

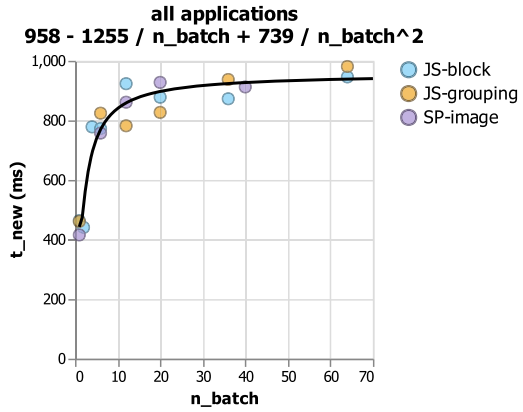

Fig. 2. The estimated model of t_{new} as a function of n_{batch} for the three applications using $y = a + b/x + c/x^2$.

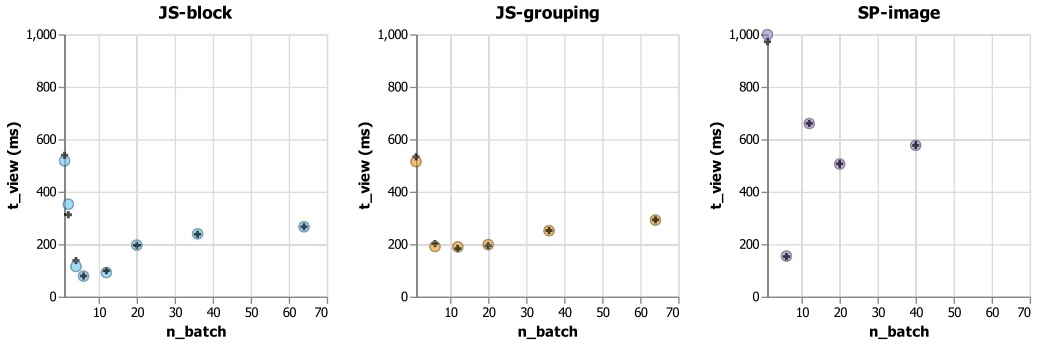

Fig. 3. t_{view} reestimation by t_{new}: The influence of reestimation to t_{view} as a function of n_{batch} for the three applications. The colored dots are initial estimations by multiple linear regression. The black crosses are reestimations by putting t_{new} back.

Procedure. With the t_{new} modeled in the last section, we reestimate t_{view} and t_{single} from the experiment data by removing the contribution of t_{new} as

$$\begin{bmatrix} t_{view} \\ t_{single} \end{bmatrix} = (X^T X)^{-1} X^T \begin{bmatrix} T_1 - N_{new,1} t_{new} \\ T_2 - N_{new,2} t_{new} \\ \cdots \\ T_n - N_{new,n} t_{new} \end{bmatrix} \tag{2}$$

where X is the N_{view} and N_{single} measured for the n trials

$$X = \begin{bmatrix} N_{view,1} & N_{single,1} \\ N_{view,2} & N_{single,2} \\ \cdots & \cdots \\ N_{view,n} & N_{single,n} \end{bmatrix}$$

and the t_{new} in the formula use modeled t_{new} values in the last section.

Outcome. Figures 3 and 4 show the influence of the reestimation on t_{view} and t_{single}.

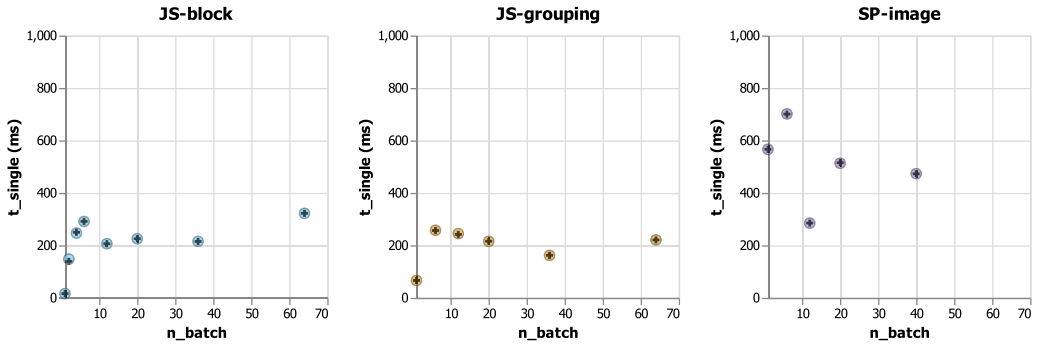

Fig. 4. t_{single} reestimation by t_{new}: The influence of reestimation to t_{single} as a function of n_{batch} for the three applications. The colored dots are initial estimations by multiple linear regression. The black crosses are reestimations by putting t_{new} back.

Comments. For both t_{view} and t_{single}, reestimation makes little change to the data points.

- **For t_{view}**: For JS-block and JS-grouping, t_{view} seems to follow a U-shaped curve. For SP-image, t_{view} is noisy and hard to interpret. The pattern of t_{view} is complex. We decide the leave t_{view} at the moment.
- **For t_{single}**: For JS-block and JS-grouping, t_{single} exhibits a constant or weak linear pattern except for the first data point.

The actions of single edit may happen during and after viewing actions. The attribution of time to single edit and view may be less accurate when few data objects are sampled (i.e., when n_{batch} is small), and the number of single edit actions is sparse.

Compared to the data points' pattern in JS-block and JS-grouping, the second and third data points of t_{single} of SP-image appear anomalous. For the SP-image example, if we ignore the second and third outlying data points, the remaining data points exhibit a constant or weak linear pattern.

It is reasonable to assume that the time cost of a single edit action (t_{single}) is reasonably consistent within an application but inconsistent across different applications. Figure 4 describes this while showing that it has a less complicated pattern than t_{view}.

The pattern of t_{single} is more straightforward than t_{view}. Thus, we decide to model it first.

B.3 Model Reestimated t_{single}

Goal. Model the reestimated t_{single} to reduce noise.

Procedure. We model t_{single} as a function of n_{batch}. There is no clear evidence that t_{single} is application-independent. Thus, we fit a model for each application. For JS-block, there are eight data points. For JS-grouping, there are six data points. For SP-image, there are five data points. We have used the Nonlinear Regression Tool [3] to produce a set of candidate model functions $t_{single} = f(n_{batch})$.

- With the number of parameters = 2, the top 3 functions are:
  - for JS-block:
    - $y = a + b/x$ with $R^2 = 0.7795$
    - $y = ae^{b/x}$ with $R^2 = 0.7049$
    - $y = \frac{ax}{x+b}$ with $R^2 = 0.6303$

  - for JS-grouping:
    - $y = a + b/x$ with $R^2 = 0.6940$
    - $y = ae^{b/x}$ with $R^2 = 0.6400$
    - $y = ax^{b/x}$ with $R^2 = 0.5938$
  - for SP-image:
    - $y = acos(bx)$ with $R^2 = 0.3560$
    - $y = a + ln(x)$ with $R^2 = 0.1385$
    - $y = ax^b$ with $R^2 = 0.1364$
- With the number of parameters = 3, the top 3 functions are:
  - for JS-block:
    - $y = \frac{1}{a+bx^c}$ with $R^2 = 0.8310$
    - $y = a + bx + c/x^2$ with $R^2 = 0.8240$
    - $y = a + b/x + c/x^2$ with $R^2 = 0.7939$
  - for JS-grouping:
    - $y = a + b/x + c/x^2$ with $R^2 = 0.9018$
    - $y = ae^{b/x+c}$ with $R^2 = 0.8927$
    - $y = ae^{b/x+cln(x)}$ with $R^2 = 0.8506$
  - for SP-image:
    - $y = ax^{bx^c}$ with $R^2 = 0.4980$
    - $y = acos(bx)$ with $R^2 = 0.3560$
    - $y = a + b/x + c/x^2$ with $R^2 = 0.3172$
- With the number of parameters = 4, the top 3 functions are:
  - for JS-block:
    - $y = a + bx + c/x + dln(x)$ with $R^2 = 0.9059$
    - $y = a + bx^{0.5} + cx + dx^{1.5}$ with $R^2 = 0.8574$
    - $y = a + bx + ((bx - c)^2 - d)^{0.5}$ with $R^2 = 0.8332$
  - for JS-grouping:
    - $y = acos(x + b) - ccos(2x + b) - dcos(3x + b)$ with $R^2 = 0.9957$
    - $y = a + b^{0.5} + cx + dx^{1.5}$ with $R^2 = 0.9804$
    - $y = a + bx + c/x + dln(x)$ with $R^2 = 0.9562$
  - for SP-image:
    - $y = a + b/x + c/x^2 + d/x^3$ with $R^2 = 0.7832$
    - $y = a + bsin(cx + d)$ with $R^2 = 0.6659$
    - $y = a + bx + c/x + d/x^2$ with $R^2 = 0.5589$

It can be seen that $y = a + b/x + c/x^2$ is reoccurring. It is the third-best for JS-block and SP-image and best for JS-grouping when #parameters = 3. Removing the c/x^2 term generates the best for JS-block and JS-grouping when #parameters = 2. Adding the d/x^3 term generates the best for SP-image when #parameters = 4.

Thus, we choose $y = a + b/x + c/x^2$ as a candidate function family for t_{single}.

Moreover, we noticed that if we ignore the 1×1 layout for JS-block and JS-grouping, t_{single} seems to follow a constant or weak linear pattern. Thus, we also choose $y = a$ and $y = a + bx$ to be candidate function families for t_{single}.

We refer to the literature for more function options. The single edit button clicking is somehow related to the decision-making task in Hick's law [2]. Hick's law suggests that the average reaction time to choose among n equally probable choices can be modeled as $T = blog_2(n + 1)$ where b is a parameter to be fitted. If we assume the solved t_{single} is a time cost for the user to point to a grid cell and click it, t_{single} may be modeled as $t_{single} = a + blog_2(n_{batch} + 1)$. The a term denotes

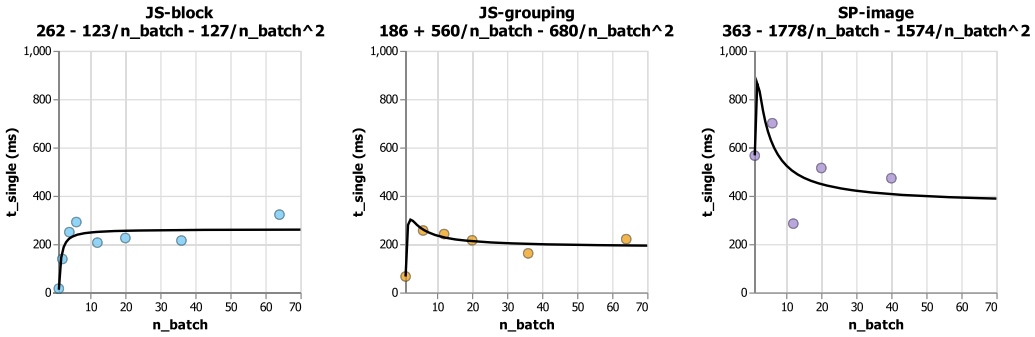

Fig. 5. The fitted t_{single} as a function of n_{batch} for JS-block, JS-grouping, and SP-image using $y = a + b/x + c/x^2$.

the constant cost of clicking, while the $blog_2(n_{batch} + 1)$ term denotes the decision time following Hick's law. Our assumption needs further investigation, as the decision time may be included in t_{view} instead of t_{single}.

In summary, we fit four models with linear regression: $y = a + b/x + c/x^2$, $y = a + blog_2(x + 1)$, $y = a$, $y = a + bx$.

Outcome.

- **For $y = a + b/x + c/x^2$**: Figure 5 shows the fitting result for t_{single}. For SP-image, due to the noisy data, the fitted model contains a spike. The functions fitted are:
  - for JS-block:
  $$t_{single} = 261.8083 - 123.2102/n_{batch} - 127.2516/n_{batch}^2$$
  with adjusted $R^2 = 0.7115, SE = 51.2691$
  - for JS-grouping:
  $$t_{single} = 186.0985 + 560.1921/n_{batch} - 680.1595/n_{batch}^2$$
  with adjusted $R^2 = 0.8363, SE = 28.4728$
  - for SP-image:
  $$t_{single} = 363.5150 - 1778.1390/n_{batch} - 1574.2689/n_{batch}^2$$
  with adjusted $R^2 = -0.3656, SE = 176.7216$
- **For $y = a + blog_2(x + 1)$**: The functions fitted are:
  - for JS-block:
  $$t_{single} = 80.7479 + 54.3526log_2(n_{batch} + 1)$$
  with adjusted $R^2 = 0.3991, SE = 73.9862$
  - for JS-grouping:
  $$t_{single} = 117.6121 + 28.5219log_2(n_{batch} + 1)$$
  with adjusted $R^2 = 0.0675, SE = 67.9599$
  - for SP-image:
  $$t_{single} = 628.0140 - 50.1844log_2(n_{batch} + 1)$$
  with adjusted $R^2 = -0.1393, SE = 161.4152$

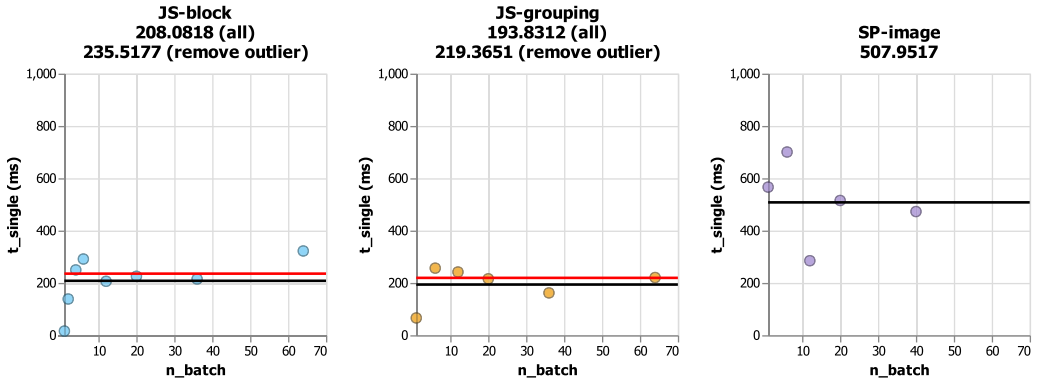

Fig. 6. The fitted t_{single} as a function of n_{batch} for JS-block, JS-grouping, and SP-image using $y = a$. The black line corresponds to the estimation without removing the outlier. The red line corresponds to the estimation after removing the outlier.

- **For y = a**: Figure 6 shows the fitting result for t_{single}. The functions fitted are:
  - for JS-block:
    - If we use all the data points:
      $$t_{single} = 208.0818$$
      with adjusted $R^2 = 0, SE = 95.4480$
    - If we remove the first data point, which seems an outlier:
      $$t_{single} = 235.5177$$
      with adjusted $R^2 = 0, SE = 60.0269$
  - for JS-grouping:
    - If we use all the data points:
      $$t_{single} = 193.8312$$
      with adjusted $R^2 = 0, SE = 70.3764$
    - If we remove the first data point, which seems an outlier:
      $$t_{single} = 219.3651$$
      with adjusted $R^2 = 0, SE = 36.0720$
  - for SP-image:
    $$t_{single} = 507.9517$$
    with adjusted $R^2 = 0, SE = 151.2289$
- **For y = a + bx**: Figure 7 shows the fitting result for t_{single}. The functions fitted are:
  - for JS-block:
    - If we use all the data points:
      $$t_{single} = 163.8567 + 2.4400 n_{batch}$$
      with adjusted $R^2 = 0.1998, SE = 85.3806$
    - If we remove the first data point, which seems an outlier:
      $$t_{single} = 205.2919 + 1.4693 n_{batch}$$
      with adjusted $R^2 = 0.1630, SE = 54.9157$

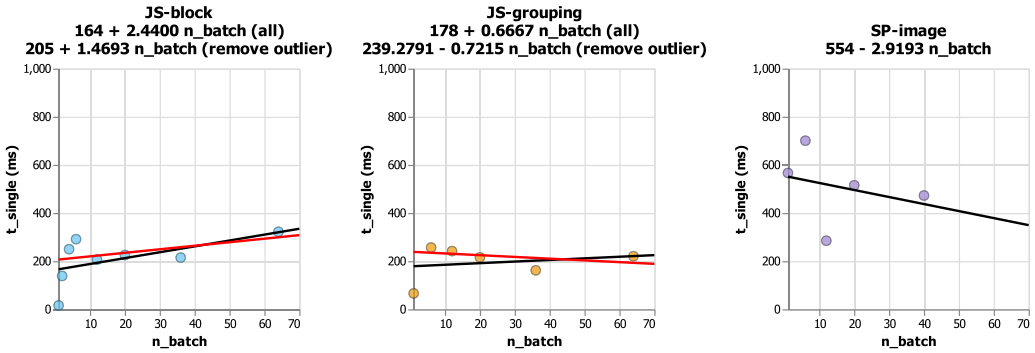

Fig. 7. The fitted t_{single} as a function of n_{batch} for JS-block, JS-grouping, and SP-image using $y = a + bx$. The black line corresponds to the estimation without removing the outlier. The red line corresponds to the estimation after removing the outlier.

—for JS-grouping:

∗ If we use all the data points:

$$t_{single} = 178.3868 + 0.6667 n_{batch}$$
$$\text{with adjusted } R^2 = -0.1882, SE = 76.7149$$

∗ If we remove the first data point, which seems to be an outlier:

$$t_{single} = 239.2791 - 0.7215 n_{batch}$$
$$\text{with adjusted } R^2 = -0.0448, SE = 36.8718$$

—for SP-image:

$$t_{single} = 554.0763 - 2.9193 n_{batch}$$
$$\text{with adjusted } R^2 = -0.2175, SE = 166.8644$$

Comments. For JS-block and JS-grouping, $y = a + b/x + c/x^2$ is the best-fitting model. For SP-image, all the models perform poorly because of the noisy and sparse samples.

For $y = a$, JS-block data fits okay with 1×1 and 8×8 being two significant outliers. As explained above, for 1×1, t_{single} approaches 0 because single edit and view are more concurrent than in other layouts. For 8×8, the large t_{single} is likely due to the latency of the interface. JS-grouping data fits well, with 1×1 being the major outlier, and the reason is the same as for JS-block. SP-image data fits poorly. Two major outliers are 2×3 and 3×4. However, note that the deviations at these two points are roughly the negation of each other. Thus, it is possible that $y = a$ suits the data.

For $y = a + bx$, for all three applications, the slopes are not too steep, and thus the analysis is similar to that of $y = a$.

Outcome. We finally decide to use the simple constant function $y = a$ because:

- It makes sense logically. When there are more grid cells, the distance between grid cells decreases. The reduced distance makes it easier to conduct single edit. Meanwhile, the size of grid cells gets smaller, which makes it hard to single edit. These two effects may offset.
- Although $y = a + b/x + c/x^2$ numerically fits JS-block and JS-grouping well, it fits SP-image poorly and thus may not be reliable.
- It is the simplest model and may avoid overfitting.

Besides, we decide not to discard the outlying points, as they do not change the resulting model much. The functions we choose are:

- **for JS-block: $t_{single} = 208.0818$**

$$\text{with adjusted } R^2 = 0, SE = 95.4480$$

- **for JS-grouping: $t_{single} = 193.8312$**

$$\text{with adjusted } R^2 = 0, SE = 70.3764$$

- **for SP-image: $t_{single} = 507.9517$**

$$\text{with adjusted } R^2 = 0, SE = 151.2289$$

B.4 Reestimate t_{view} by t_{new} and t_{single} Models

Goal. Reestimate t_{view} by removing the previous modeled t_{new} and t_{single} from the equations to possibly reduce the noise and make sure the original equations still approximately hold.

Procedure. With the t_{new} and t_{single} modeled in the last sections, we reestimate t_{view} from the experiment data by removing the contribution of t_{new} and t_{single} to equations as

$$\left[t_{view}\right] = (X^T X)^{-1} X^T \begin{bmatrix} T_1 - N_{new,1} t_{new} - N_{single,1} t_{single} \\ T_2 - N_{new,2} t_{new} - N_{single,2} t_{single} \\ \cdots \\ T_n - N_{new,n} t_{new} - N_{single,n} t_{single} \end{bmatrix} \tag{3}$$

where X is the N_{view} measured for the n trials

$$X = \begin{bmatrix} N_{view,1} \\ N_{view,2} \\ \cdots \\ N_{view,n} \end{bmatrix}$$

and the t_{new} and t_{single} in the formula use modeled t_{new} and t_{single} values in the last sections.

Outcome. Figure 8 shows the influence of the reestimation on t_{view}.

Comments. The modeling of t_{new} does not significantly change the estimation of t_{view}. By comparison, the additional modeling of t_{single} changes t_{view} observably.

For JS-block, the 1×1 point lowers significantly after reestimation for t_{new} and t_{single} because the modeled t_{single} at 1×1 is much larger than the initial t_{single} which is close to 0. The reestimation decreases the t_{view} at 1×1 to compensate for this change.

Similarly, for JS-grouping, the 1×1 point lowers significantly. For SP-image, the 2×3 point increases significantly while the 3×4 point decreases significantly.

All the other data points are hardly changed.

B.5 Model Reestimated t_{view}

Goal. Model the reestimated t_{view} to reduce noise.

Procedure. We model t_{view} as a function of n_{batch}. We fit a model for each application because we expect viewing time to be related to the content. For JS-block, there are eight data points.

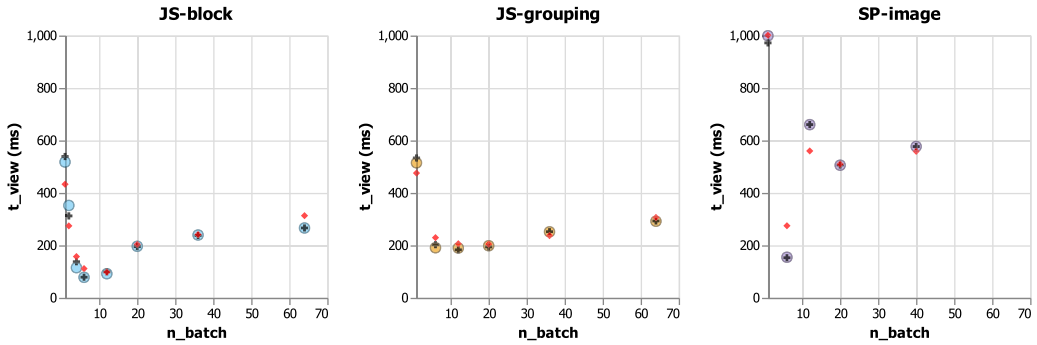

Fig. 8. t_{view} reestimation by t_{new} and t_{single}: The influence of reestimation to t_{view} as a function of n_{batch} for the three applications. The colored dots are initial estimations by multiple linear regression. The black crosses are reestimations by putting only t_{new} back. The red diamonds are reestimations by putting both t_{new} and t_{single} back.

For JS-grouping, there are 6 data points. For SP-image, there are five data points. We assume the suitable model function family should be shared among applications. We have used the Nonlinear Regression Tool [3] to produce a set of candidate model functions $t_{new} = f(n_{batch})$.

- With the number of parameters = 2, the top 3 functions are:
  - for JS-block:
    - $y = ax^{b/x}$ with $R^2 = 0.5548$
    - $y = \frac{ax}{x+b}$ with $R^2 = 0.5397$
    - $y = ae^{b/x}$ with $R^2 = 0.5098$
  - for JS-grouping:
    - $y = \frac{ax}{x+b}$ with $R^2 = 0.8494$
    - $y = ae^{b/x}$ with $R^2 = 0.8477$
    - $y = a + b/x$ with $R^2 = 0.8174$
  - for SP-image:
    - $y = ax^{b/x}$ with $R^2 = 0.8815$
    - $y = \frac{ax}{x+b}$ with $R^2 = 0.7500$
    - $y = ae^{b/x}$ with $R^2 = 0.7289$
- With the number of parameters = 3, the top 3 functions are:
  - for JS-block:
    - $y = a + bx + c/x$ with $R^2 = 0.9310$
    - $y = x^a e^{bx^c}$ with $R^2 = 0.9093$
    - $y = a + bx + c/x^2$ with $R^2 = 0.8822$
  - for JS-grouping:
    - $y = a + bx + c/x$ with $R^2 = 0.9958$
    - $y = x^a e^{bx^c}$ with $R^2 = 0.9886$
    - $y = ax^b e^{cx}$ with $R^2 = 0.9874$
  - for SP-image:
    - $y = a + \frac{b}{x+c}$ with $R^2 = 0.9803$
    - $y = ae^{\frac{b}{x+c}}$ with $R^2 = 0.9795$
    - $y = a + b/x + c/x^2$ with $R^2 = 0.9546$

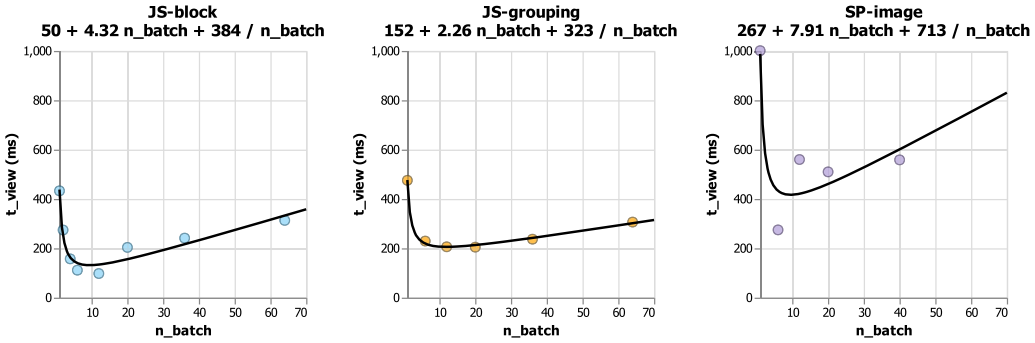

Fig. 9. The fitted t_{view} as a function of n_{batch} for the three applications using $y = a + bx + c/x$.

- With the number of parameters = 4, the top 3 functions are:
  - for JS-block:
    - $y = a + bx + c/x + dln(x)$ with $R^2 = 0.9378$
    - $y = a + bx + c/x + d/x^2$ with $R^2 = 0.9311$
    - $y = a + bx + c/x$ with $R^2 = 0.9310$
  - for JS-grouping:
    - $y = a + bx + c/x + dln(x)$ with $R^2 = 0.9998$
    - $y = a + bx + c/x + d/x^2$ with $R^2 = 0.9997$
    - $y = ae^{b/x} + ce^{d/x}$ with $R^2 = 0.9993$
  - for SP-image:
    - $y = \frac{a+bx+cx^2}{x+d}$ with $R^2 = 0.9862$
    - $y = a + \frac{b}{x+c}$ with $R^2 = 0.9803$
    - $y = ae^{\frac{b}{x+c}}$ with $R^2 = 0.9795$

It can be seen that $y = a + bx + c/x$ is reoccurring. It is the best for JS-block and JS-grouping when #parameters = 3 and the third-best for JS-block when #parameters = 4. Adding $dln(x)$ and d/x^2 terms generate the best and second best functions for JS-block and JS-grouping when #parameters = 4. Removing the cx term generates the third-best for JS-grouping when #parameters = 2. For SP-image, the data is sparse and noisy and thus not decisive.

Thus, we choose $y = a + bx + c/x$ as the function family for t_{view}. We fit the model with linear regression.

Outcome. Figure 9 shows the fitting result for t_{view}. The functions fitted are:

- **for JS-block:** $\mathbf{t_{view} = 50.4283 + 4.3202 n_{batch} + \frac{383.5033}{n_{batch}}}$
  with adjusted $R^2 = 0.9034, SE = 34.8005$
- **for JS-grouping:** $\mathbf{t_{view} = 151.9951 + 2.2618 n_{batch} + \frac{322.6178}{n_{batch}}}$
  with adjusted $R^2 = 0.9930, SE = 8.7040$
- **for SP-image:** $\mathbf{t_{view} = 267.0109 + 7.9103 n_{batch} + \frac{712.8504}{n_{batch}}}$
  with adjusted $R^2 = 0.6472, SE = 156.1362$

Table 4. Modeled Operator Time Costs: The Final Estimations of Operator Time Costs

Application	Layout	t_{new} (ms)	t_{view} (ms)	t_{single} (ms)
JS-block	1 × 1	443	438	208
JS-block	1 × 2	515	251	208
JS-block	2 × 2	691	164	208
JS-block	2 × 3	769	140	208
JS-block	3 × 4	859	134	208
JS-block	4 × 5	897	156	208
JS-block	6 × 6	924	217	208
JS-block	8 × 8	939	333	208
JS-grouping	1 × 1	443	477	194
JS-grouping	2 × 3	769	219	194
JS-grouping	3 × 4	859	206	194
JS-grouping	4 × 5	897	213	194
JS-grouping	6 × 6	924	242	194
JS-grouping	8 × 8	939	302	194
SP-image	1 × 1	443	988	508
SP-image	2 × 3	769	433	508
SP-image	3 × 4	859	421	508
SP-image	4 × 5	897	461	508
SP-image	5 × 8	927	601	508

Comments. The data clearly shows a U-shaped pattern. We conjecture that small grid cells require more attention and time to see clearly, while large grid cells can be distracting and require more eye movement to view one object.

$n_{batch} \cdot gridCellArea = interfaceArea$ is a constant. If we assume the penalty of time cost to look into details of small objects to be inversely proportional to object area, the bx term can be explained. Assuming the eye moment time to scan an object is proportional to the object's area, the c/x term can be explained. The viewing action may contain some reaction latency of the human. Thus, the a term can be explained.

B.6 Final Estimations of Operator Time Costs

Table 4 shows the final estimations of operator time costs with the models we have fitted for t_{new}, t_{view}, and t_{single}.

REFERENCES

[1] Paul M. Fitts. 1992. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology: General* 121, 3 (1992), 262–269.

[2] William Edmund Hick. 1952. On the rate of gain of information. *The Quarterly Journal of Experimental Psychology* 4, 1 (March 1952), 11–26.

[3] Xuru. 2006. Online Regression Tools. Accessed on Apr 20, 2020.