Validation of visual inference methods in statistics by use of deep learning

Anne Helby Petersen[1] & Claus Thorn Ekstrøm
Section of Biostatistics, Department of Public Health, University of Copenhagen, Denmark

Abstract

Visual inference is used for various tasks in statistics – including model diagnostics and exploratory data analysis – and though attractive due to its intuitive nature, the lack of available methods for validating plots is a major drawback. We propose a new validation method for visual inference. We use deep neural networks to distinguish between plots simulated under two different data generating mechanisms (null or alternative), and report the classification accuracy as a technical validation score (TVS). The TVS measures the information content in the plots, and TVS values can be used to compare different plots or different data generating mechanisms, thereby providing a meaningful scale that new visual inference procedures can be validated against. We apply the method to three popular diagnostic plots for linear regression, the scatter plot, the quantile-quantile plot and the residual plot, and we find the TVS to be an informative measure of validity.

Keywords: graphical inference; linear regression; neural network; model diagnostics; visualization

1. Introduction

Visual inference methods in statistics comprise a wide – and ever expanding – range of different visualization methods, or *plots*, that are useful tools for model diagnostics, explorative data analysis, and robust statistical inference. However, plots are intrinsically difficult to validate themselves: Both traditional mathematical-analytical methods and modern simulation-based methods for validating statistical objects fall short when the statistical object is a plot.

Amazon's Mechanical Turk platform [1] has been used to conduct human subject experiments for different topics within visual inference validation, including design choices [2, 3, 4], validation of new plot methods for comparing differences in distributions [5], and case-studies for well-known diagnostic plots for specific statistical models such as linear mixed-effects models [6] and variable selection for linear regression [7]. However, such experiments have a number of limitations. First, if the plot is intended to be used by highly specialized experts, such as empirical researchers within a certain field, it may simply not be possible to recruit adequate participants through platforms such as Mechanical Turk. But without recruitment online, human subject experiments quickly become very time-consuming and costly to conduct. Secondly, even if less specialized participants are needed, the platform has limited geographical reach [8] which may pose challenges to generalizations across cultures and languages. Thirdly, for many statistical plots, the relevant validity concern is not whether a person can immediately and intuitively interpret the plot correctly, but rather whether a person can realistically be trained to do so: We do not expect students to be able to correctly interpret residual plots before teaching them about the underlying statistical concepts. The experimental setup on Mechanical Turk is thus quite artificial as it mainly facilitates to measure the degree to which participants can intuitively interpret plots correctly.

All in all, the lack of a generally applicable method for validating new plots poses severe limitations to adoption of new visual inference methods and thereby stalls an otherwise useful avenue of new methodological research.

In this article, we address this limitation by proposing a new method for validating statistical plots that are designed to inform a binary decision, for example whether a statistical model is misspecified or not. Our method trains deep neural networks to classify diagnostics plots as problematic or not, and we interpret the performance of these neural networks as an indication of the potential information contained in the plot. If the neural network cannot be trained to classify the plots well, we claim that it is unlikely that humans will be able to do so. On the other hand, if it is possible for the neural network to classify the plots sufficiently

[1]Correspondence: ahpe@sund.ku.dk

well, then the plots do hold the information necessary for the task at hand and hence it may also be possible for humans to learn to classify them correctly. In this sense, our method tests a necessary, but not sufficient, validation criterion for visual inference methods. Our validation method is therefore not intended to replace human subject experiments, but instead to complement the experiments by providing a measure of whether a visual inference task is at all feasible before moving on to the more time-consuming, costly and difficult question of whether it is *useful* for human subjects.

We showcase the validation method on three classic textbook diagnostic plots for linear normal regression, namely the scatter plot, the residual plot and the QQ plot. We train neural networks to identify plots suffering from three different types of misspecification: a misspecified mean structure, a misspecified variance structure, or both. Moreover, we investigate how our validation method varies with the degree of misspecifiation and the within-plot sample size. Thus, we provide a thorough investigation into the potential of our proposed visual inference validation method in a rather well-known statistical scenario, and hence, our results may be used as a benchmark for validation of other visual inference methods.

The article is structured as follows: First, in Section 2 we introduce the statistical problem that we consider in this article, namely making binary decisions based on visual inference methods. In Section 3, we present our proposal for how visual inference methods may be validated by use of deep learning. We apply the method to validate diagnostic plots for linear normal regression in Section 4. Finally, we discuss strengths and limitations of the proposed method in Section 5.

2. Binary decision making in visual inference

Visual inference in statistics is concerned with several topics, including exploratory data analysis and model diagnostics. Moreover, insights drawn from visual statistical objects, such as plots, can be both qualitative and quantitative of nature. In this article, we will focus only on a subset of visual inference tasks, namely binary decisions akin to statistical testing. We aim to provide a novel method for validating methods that use visual inference to deliver a binary decision.

Formally, we can define the problem as follows: Let X_0 and X_{alt} be two observed datasets consisting each of n observations of p random variables, and let f_0 and f_{alt} denote their respective data generating mechanisms. Moreover, let g be a plotting function, that is, a function that takes an observed dataset and creates a plot. We then consider the following question: If we only see unlabeled examples of $g_0 := g(X_0)$ and $g_{\text{alt}} := g(X_{\text{alt}})$, how well can determine which plots were created using data from f_0 and which plots used data from f_{alt}? In other words, do the plots hold enough information about the difference between f_0 and f_{alt} to tell the two apart? Or similarly: Is the difference between f_0 and f_{alt} large enough to be detectable in a visualization using g?
Example: As an example, consider the two scatter plots in Figure 1, each annotated with a linear regression line. One depicts observations from the data generating mechanism

$$f_0(X, \epsilon) = X + \epsilon$$

while the other shows observations from

$$f_{\text{alt}}(X, \epsilon) = \exp(X) + \epsilon$$

In both cases, X and ϵ are independent standard normal random variables, and the plots depict $n = 50$ independent observations that were standardized before plotting. The plotting function can informally be described as

$$g_{\text{scatter}}(x, y) = \text{draw an x-y scatter plot and annotate it with a regression line}$$

Can we tell which plot corresponds to which data generating mechanism? In this scenario, most scholars with a basic knowledge about mathematical functions will be able to conclude that plot (a) corresponds for f_0 while the (b) plot corresponds to f_{alt} by noting the curvature in the latter plot.

3. Validation of binary visual inference methods

Given a choice of two data generating mechanisms, f_0 and f_{alt}, whether or not a specific plot function, g, can be used to distinguish between the two is a question consisting of two fairly unrelated components, namely:

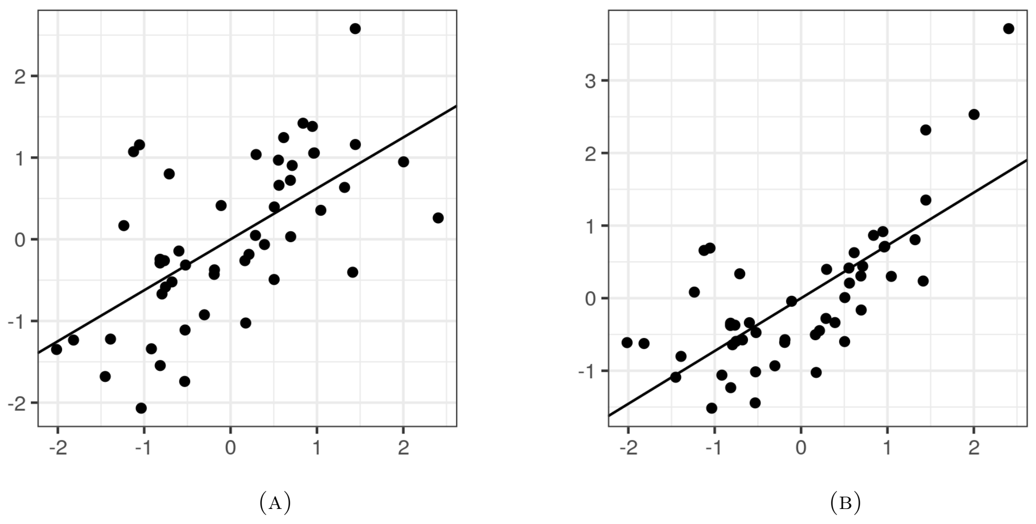

FIGURE 1. Scatter plots depicting $x{-}y$ pairs from two different data generating mechanisms, namely $f_0 : Y = X + \epsilon$ (a) and $f_{\text{alt}} : Y = \exp(X) + \epsilon$ (b). Note that the variables were each standardized prior to plotting. The same random seed was used for simulating data for both plots.

(1) Does g convey enough information about f_0 and f_{alt} to make it *technically* possible to tell them apart? If so, we will refer to the plotting function as *technically valid* for distinguishing between f_0 and f_{alt}.
(2) Is it feasible to train human inspectors to be able to tell examples of g_0 and g_{alt} apart? If so, we will refer to the plot function as *practically valid* for distinguishing between f_0 and f_{alt}.

In this article we address only the first question. Moreover, rather than providing a yes or no answer concerning technical validity, we propose to measure to what degree technical validity is fulfilled and to do so in a manner that makes it possible to compare technical validity across different plot functions, different combinations of data generating mechanisms, or both.

Note that both types of validity of course depends on the choices of f_0 and f_{alt}; if they are very similar - or even equal - then no plot will make it possible to tell them apart, and in this case, we would like any validity score to be low.

We propose to use the following method, consisting of three consecutive steps, for measuring the *technical validity* of a plot function g for distinguishing between two data generating mechanisms f_0 and f_{alt}:

1. **Simulate plots::** Simulate a larger number of instances of plots from each data generating mechanism. Randomly allocate a proportion of the plots - along with labels indicating which data generating mechanism created them - to training data, while withholding the remaining plots and their labels for validation data.
2. **Train model::** Use the training data and training labels to construct a classification function h, which takes in an unlabeled (new) plot g and returns one of two classes, *null* or *alt*.
3. **Score model::** Compute a *technical validity score* by comparing the actual labels on the validation data with the labels produced by applying h to the validation plots.

Below, we discuss each of the three steps in more detail and provide suggestions for how they may be implemented.

3.1. Simulating plots.

Let $n \in \mathbb{N}$ be the sample size for each plot, $M \in \mathbb{N}$ be the number of simulation steps, and let $r \in (0, 100)$ be the percentage of data that will be allocated to training data. Define $n_{\text{train}} = \text{floor}(\frac{r}{100} \cdot M)$ and $n_{\text{val}} = M - n_{\text{train}}$ as the number of plots in the training data and validation data, respectively. For each $m \in \{1, ..., M\}$, do the following:

(1) Use f_0 to simulate n independent observations, x_0^m, and produce a plot $g_0^m := g(x_0^m)$.
(2) Use f_{alt} to simulate n independent observations, x_{alt}^m, and produce a plot $g_{\text{alt}}^m := g(x_{\text{alt}}^m)$.

Without loss of generality, we can then construct the training data as the first n_{train} plots from each data generating mechanism,

$$G_{\text{train}} = \{g_0^1, ..., g_0^{n_{\text{train}}}, g_{\text{alt}}^1, ..., g_{\text{alt}}^{n_{\text{train}}}\}$$

and the training labels as a vector of their corresponding data generating mechanism,

$$Y_{\text{train}} = \{\text{null}, ..., \text{null}, \text{alt}, ..., \text{alt}\}$$

where each labeled is repeated n_{train} times. In a similar fashion, we construct the validation data as

$$G_{\text{val}} = \{g_0^{n_{\text{train}}+1}, ..., g_0^M, g_{\text{alt}}^{n_{\text{train}}+1}, ..., g_{\text{alt}}^M\}$$

and their corresponding labels as

$$Y_{\text{val}} = \{\text{null}, ..., \text{null}, \text{alt}, ..., \text{alt}\}$$

where each labeled is repeated n_{val} times.

Exactly how to simulate from f_0 and f_{alt} of course depends on the specific data generating mechanisms, but we believe that most mechanisms that would be of interest to study using visual inference can be simulated using Monte Carlo methods (see for example [9]).

3.2. **Training the model.** There are many possible ways to construct a classification function $h : g \mapsto \{\text{null}, \text{alt}\}$ from the training data, and it is very likely that different methods will give different performances, depending on the choice of g, f_0, f_{alt} and n. Instead of trying to find an optimal model for each specific plot validation problem, we propose to use a general solution that should be expected to perform reasonably well on most tasks: We construct h by use of a deep neural network.

A deep neural network is a highly flexible machine learning model that is particularly useful for image classification. Deep neural networks are *universal approximators* which implies that they can be used to approximate any classification function [10]. The specific configuration of the neural network model is typically refered to as its *architecture*. The parameters for the classification function are typically denoted *weights* and they are estimated by minimizing a loss function using a recursive numerical optimization method such as stochastic gradient descent [11]. For this optimization procedure, the training data can be considered in *batches*, thereby allowing for quicker (but coarser) optimization steps that may be computed in parallel, while also economizing computer memory needed to open the data (often consisting of image files). The optimization is carried on for a number of *epochs*, where each epoch marks that all training data have been considered in batches. Thus, the smaller the batches and the larger the number of epochs, the longer the optimization procedure will take, and the more it can learn from the training data. At some point, however, the model will overfit to the training data, thus producing a classification function that does not perform well on new validation data.

We propose to use the so-called *Inception* architecture for a neural network, which has outperformed all other models in the ImageNet image classification challenge [12]. The ImageNet challenge is generally considered the standard benchmark metric for image classification, as well as some other computer vision tasks [13]. The original task of the challenge was to classify images from a specific dataset into 1000 known labels with as large validation accuracy as possible. The images are photographs of a variety of everyday objects, animals, plants and more. The image classification task was last included in the challenge in 2014, where the Inception network achieved a classification error of only 6.67%. This error rate is comparable with human precision on the same task [13].

Even though the Inception network has been designed to classify images that are generally rather different from statistical plots – depicting for example cats, trucks or books – we conjecture that it will still be useful for most visual inference tasks simply because the architecture is so flexible that it will be able to learn structures from most types of images – including statistical plots. The only modification we apply to the network is in terms of its *input* and *output layers*, that is, the structure that is expected of the inputted training data and the outputted class probabilities. We modify the input layer to fit the dimensions of the plots we are considering, and we modify the output layer to classify into only two classes (*null* or *alt*) rather than the 1000 possible classes available for the ImageNet challenge. We do not use pretrained starting weights, that is, we do not give the optimization a warm start by starting the algorithm with weights computed from for example the ImageNet challenge.

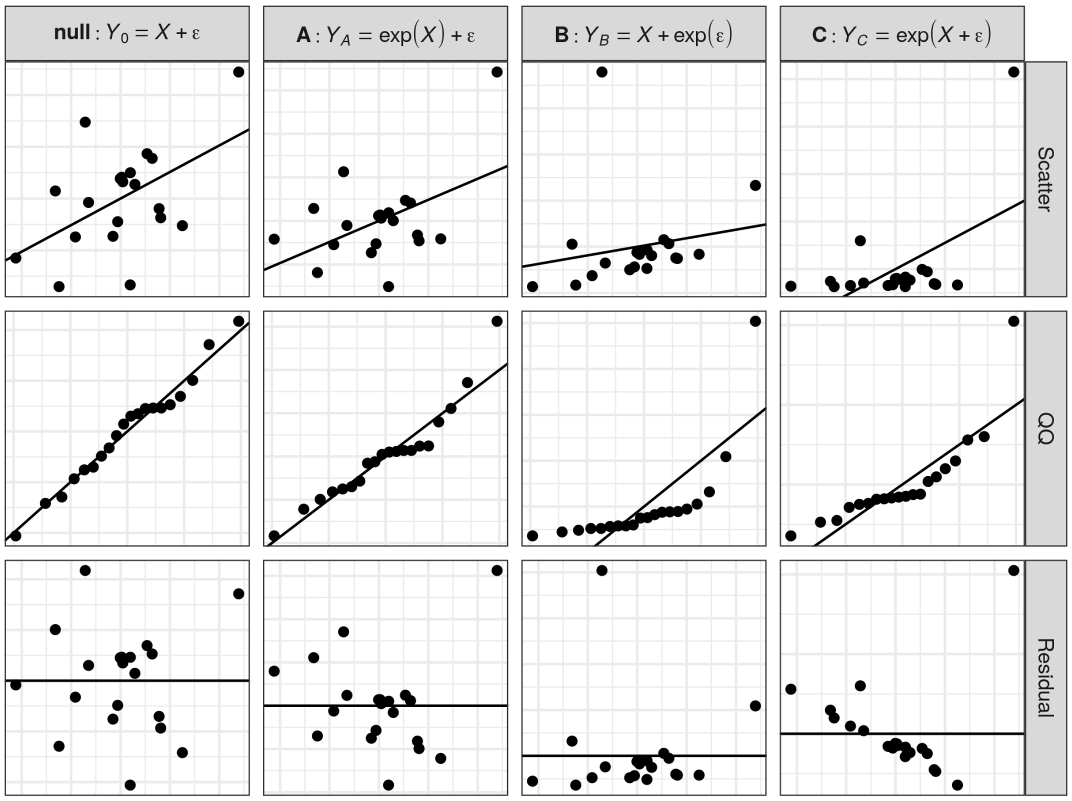

FIGURE 2. Examples of plots for each combination of data generating mechanism and choice of plot function. Axis scales have been removed for clarity here, but other than that, the examples shown here have the same design as used for training the network (including the annotated straight lines). We refer to Figure 1 for an example of the exact plot design used for training the model. The same random seed has been used to simulate data for all the 12 plots shown here and we have used $\mathrm{V}(X) = \mathrm{V}(\epsilon) = 1$ and $n = 20$ for simulating each data set.

3.3. **Computing the technical validity score.** We propose to use the validation data accuracy as a technical validity score. Hence, we define

$$\mathrm{TVS}(g, f_0, f_{\mathrm{alt}}, n) = \frac{1}{n_{\mathrm{val}}} \sum_{i=1}^{n_{\mathrm{val}}} 1(h(G_{\mathrm{val}}[i]) = Y_{\mathrm{val}}[i])$$

where $1(*)$ denotes the indicator function.

In order to not make the result too sensitive to the number of epochs used to train the model, we do not just report the TVS obtained after the final epoch. Instead we compute the TVS after each epoch is completed, and use the 80th percentile of the TVS values over all epochs as the final TVS.

The notation for the TVS is intended to stress that it is a validation measure for a specific combination of plot function (g), data generating mechanisms (f_0 and f_{alt}) and within-plot sample size (n).

Clearly, the TVS will also depend on the amount of available training data, the number of epochs and the batch sizes used for training the model. All these factors will affect the classification function, h. We therefore advice that TVS values should only be compared for similar protocols for constructing h. This implies using the same network architecture (for example *Inception*), the same number of simulated plots for each mechanism (M), the same batch size and the same number of epochs.

4. Application: Model diagnostics for linear normal regression

In order to showcase our proposed method for validating plots, we examine the validity of a selection of diagnostic plots that are often recommended for assessing whether the assumptions underlying linear normal regression are fulfilled [14]. Figure 2 provides four examples for each diagnostic plot, one created under a null data generating mechanism, where the linear normal regression model is well-specified, and three created under varying types of misspecification. For each of these three types of misspecification, we will investigate how well the deep neural network can distinguish between plots from the misspecificied model and plots from a null model that is well-specified. We will compute a TVS for each of these three plots in order to determine whether they are each useful for identifying model misspecification in linear normal regression. This will give us comparable numbers so that we can not only discuss whether each plot is valid for the task or not, but also which one is the most useful for identifying misspecification.

4.1. **Plotting functions.** We consider three different diagnostic plots for linear normal regression:

Scatter plot:: A standard xy scatter plot. We add an estimated regression line from the linear normal regression model (1).

QQ plot:: A quantile-quantile (QQ) plot comparing standardized residuals from (1) with the standard normal distribution. We annotate the plot with a straight line with intercept 0 and slope 1.

Residual plot:: A scatter plot with x values on the horizontal axis and standardized residuals from (1) on the vertical axis. The plot is annotated with a horizontal line with intercept 0.

The plots are annotated with axis scales, see Figure 1 for an example of the exact plot design for the scatter plots. We do not enforce standardized scaling of the axes, but instead allow the plotting software (further details below) to make this choice automatically. We believe this makes the results correspond more realistically with how plotting software is used in practice. This means that the y-axes are potentially scaled quite differently from plot to plot. Note that the x-axes will always be on approximately the same scale, since the x values are always draws from a normal distribution, and they are always scaled to have mean 0 and variance 1.

4.2. **Data generating mechanisms.** Let $X \sim N(0, \sigma^2)$ and $\epsilon \sim N(0,1)$ be independent. We then consider the null data generating mechanism,

$$Y_0 = f_0(X, \epsilon) := X + \epsilon$$

and note that linear normal regression of realizations of Y on realizations of X will then be well-specified as long as the realizations (observations) are independent. We wish to validate whether diagnostic plots can be used to identify deviations from the assumptions underlying this model by considering three different possible alternative data generating mechanisms:

$$Y_A = f_{\text{alt}}^A(X, \epsilon) := \exp(X) + \epsilon$$
$$Y_B = f_{\text{alt}}^B(X, \epsilon) := X + \exp(\epsilon)$$
$$Y_C = f_{\text{alt}}^C(X, \epsilon) := \exp(X + \epsilon)$$

We fit linear normal regression models

$$y_i = \alpha + \beta \cdot x_i + e_i \tag{1}$$

where the e_is are independent error terms and y_i are drawn from one of f_0, f_{alt}^A, f_{alt}^B and f_{alt}^C. In the latter three scenarios, the model will be misspecified. For f_{alt}^A the mean structure is misspecifed, for f_{alt}^B the variance structure is misspecified, and for f_{alt}^C both the mean and variance structures are specified simultaneously. Note that misspecification due to f_{alt}^C is the familiar setting from statistics textbooks where analysts would be advised to fit regression models on a logarithmic transformation of y rather than the untransformed y variable.

As mentioned above, the validity of a visual inference method will often depend on the within-plot sample size. We consider four possible values of n, namely $n \in \{5, 10, 20, 50\}$.

Moreover, we expect the validity of the diagnostic plots to also depend on the degree of misspecification. We control this factor through the variance of X, which we allow to take three different values, $\sigma^2 \in \{0.5, 1, 2\}$, while $V(\epsilon) = 1$ remains fixed. Which setting of σ corresponds to the most misspecification will vary for the different data generating mechanisms. For mechanism A, large values of σ will imply larger curvature in the mean and hence a larger degree of misspecification. For mechanism B, varying σ should not impact

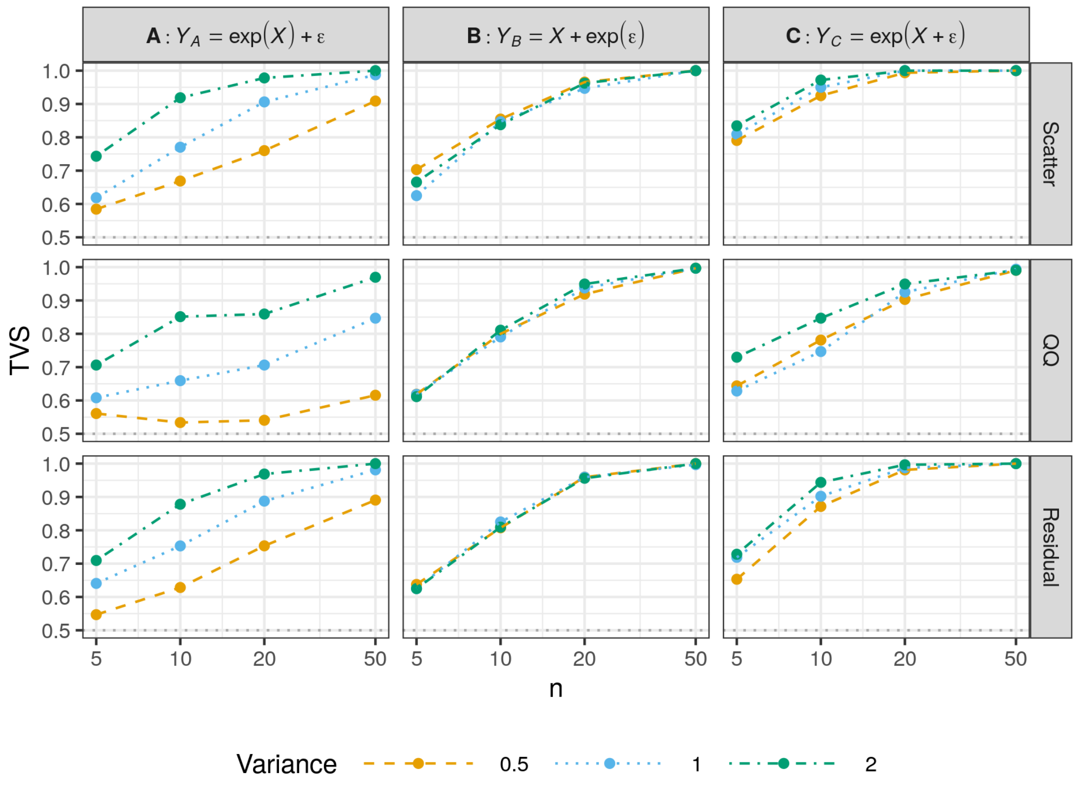

FIGURE 3. TVS values for each combination of alternative data generating mechanism, plot function, sample size and σ^2. Note that the horizontal axis has been logarithmically scaled. The plots are annotated with a horizontal dotted line marking a TVS of 0.5, which corresponds to random guessing, which is equivalent with there being no information available in the plot for distinguishing between f_0 and f_{alt}.

the degree of misspecification, as the misspecified part of the model is not directly affected (see details in Appendix A). For mechanism C, the degree of misspecification will vary with σ similarly to mechanism A. However, whether this will have an effect on the difficulty of the classification task will most likely depend on which plot is used and whether the plot is more sensitive to mean or variance structure misspecification.

4.3. **Implementation details.** We consider all possible combinations of plot (scatter, QQ, residual), alternative data generating mechanism (A, B, C), sample size (5, 10, 20, 50) and variance of X (0.5, 1, 2). This gives us a total of 108 different scenarios to validate.

We simulate $B = 5000$ null plots, and we simulate B alternative plots for each alternative data generating mechanism. We allocate $r = 90$ % of the plots to training data, resulting in $n_{\text{train}} = 9000$ and $n_{\text{val}} = 1000$ for each scenario.

We use the *InceptionV3* version of the Inception network, and train it using 20 epochs with batches of 100 plots. We compute the TVS by the end of each epoch and report the 80th percentile of the TVS values across epochs in order to get a stable measure of possible performance.

In the interest of computer memory, we produce the plots as small PNG image files of size 5×5 cm and a resolution of 72 *dots per inch* (DPI). The full plot collection is avaliable online at `http://publicifsv.sund.ku.dk/~ahp/visval/visvalplots.zip`.

We use R [15] to generate plots and perform all computations. The plots are produced using the *ggplot2* package [16] with parallel computations by use of the *foreach* [17] and *doMC* [18] packages. The neural networks are trained using the *keras* package [19]. All code is available online at `https://github.com/annennenne/visval`.

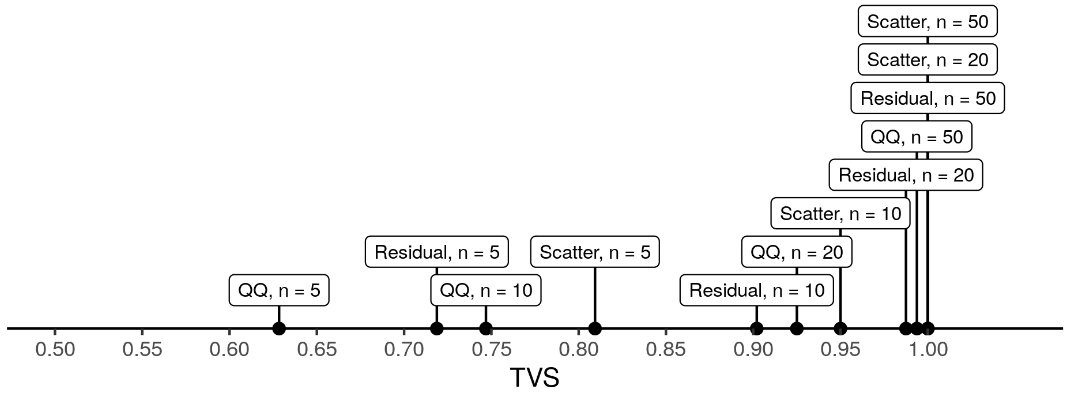

FIGURE 4. TVS values for identifying simultaneous mean and variance misspecification (mechanism C) with σ^2.

4.4. **Results.** Figure 3 provides an overview of the TVSs for each validation scenario. The obtained TVSs vary between 0.53 (for QQ plots with $n = 10$, $\sigma^2 = 0.5$ and data generating mechanism A) and 1.00 (obtained for several combinations, most of which have $n = 50$). Note that a TVS of 0.5 corresponds to random guessing, while a TVS of 1.0 describes perfect classification. The results in Figure 3 thus show that our proposed validation method is able to produce TVS values that span almost the full meaningful range from 0.5 to 1.

With the exception of a single scenario for the QQ plot, we see that the TVSs increase with increasing within-plot sample size (n), which corresponds well with what should be expected of a validation score: If there are more observations in the plots, it should be easier to tell apart the data generating mechanisms. Hence, the positive association between n and the TVS shows that our method is in fact measuring a meaningful characteristic of the plots.

With respect to the degree of misspecification, as controlled through σ, we also find what we expected: For mechanism A, we see that larger values of σ, that is, larger degrees of misspecification, produce easier classification tasks and hence larger TVS values. For mechanism B, we see no impact of σ on the TVS, as expected. For mechanism C, we find some influence of σ, especially for low values of n and more so for the QQ and residual plots than for the scatter plot. This suggests that in these scenarios, the plots are more sensitive to mean misspecifiation than to variance misspecification.

Comparing the three different plots, we find that the scatter plot generally obtains the highest values of TVS for almost all combinations of misspecification, sample size and σ^2. The only exceptions are the combinations $\{n = 5, \sigma^2 = 1\}$ for mechanism A and B, as well as the combination $\{n = 20, \sigma^2 = 1\}$ for mechanism B, and in all these three scenarios, the QQ plot slightly outperforms the others. But for the remaining 105 scenarios, the scatter plot performs better, or equally well as, the others. The residual plot is in a close runner-up for most scenarios with very similar performance to that of the scatter plot.

An interesting finding concerning the performance of the scatter plot is that it is only good at detecting mean structure misspecification (mechanism A), which it is typically used for, but it also performs better or equally well as its compepitors for variance structure misspecification detection, even though the QQ plot and the residual plot were designed specifically with this purpose in mind. Hence, at least for the specific versions of mean and variance structure misspecification considered here, our results suggest that it may be sufficient to only consider scatter plots for linear normal regression model diagnostics. Human subject experiments can be used to confirm whether this holds true not only for the technical validity of the plots, but also for the practical validity.

The conclusion for the QQ plot is not as uplifting. This plot performs varyingly, and at times rather poorly. It generally needs a larger within-plot sample size to achieve the same TVS as the other plots, and hence it does not make as good use of the information in the plots. Moreover, especially for mechanism A, the QQ plot is markedly more sensitive towards the choice of σ than the other plots: In this scenario, we see large differences between the TVS values for the QQ plot for different choices of σ, especially for large sample sizes (see the first column of Figure 3). Even with $n = 50$, the QQ plot achieves very poor performance for $\sigma^2 \in \{0.5, 1\}$, as compared to the other plots. For these reasons, we consider the technical validity of the QQ plot to be overall lower compared to its competitors.

The results from this application may be used to form a calibration scale for future visual inference validity investigations using the same methodology. Figure 4 provides an overview of the performances of the three plots for identifying simultaneous variance and mean structure misspecification (mechanism C) with $\sigma^2 = 1$. The calibration scale in Figure 4 is intended to be used as follows: If, for example, the validation of a new plot results in a TVS of 0.8, we can use the calibration scale to conclude that the plot is more valid than a QQ plot with 10 observations, but less valid than a scatter plot with 5 observations. We believe that statements like this are easily interpretable and that they can provide a straight forward and intuitive way to evaluate and describe the performance of visual inference methods.

5. Discussion

We have proposed to use deep neural networks for validating and comparing visual inference methods. We have showcased our validation method in an application considering three different diagnostic plots combined with three different types of misspecifiation for linear normal regression.

The central idea behind our validation method, namely measuring the validtiy of a plot by determining if null and alternative plots can be told apart, has been applied in previous research as well. [20] have proposed to use the so-called *line-up protocol* to systematically draw proper statistical inference from visual plots. In the line-up protocol, a "true" data plot is randomly inserted among a number of null-plots (produced for example by use of random permutations), and the investigator then needs to identify which plot corresponds to the true data. A similar setup has been proposed by [21], albeit with a different purpose in mind. Here, a data plot is again randomly placed among a number of null plots, but the procedure is intended as a learning tool, rather than a road to statistical inference: By providing an investigator with not only the data plot, but also a selection of null plots, it may help her in interpreting the data plot correctly and especially in not mistaken random noise for signal.

A limitation of the proposed method is that it is only suitable for visual inference methods concerned with binary decision making. While the method could easily be extended to accommodate decisions with a larger number of possible alternative answers, simply by allowing for more categories in the output layer of the neural network, it is not very useful for visual inference that calls for qualitative statements or interpretation. In such scenarios, technical validity is difficult to define and hence an entirely different approach will be needed.

Moreover, a large TVS of course does not guarantee that the plot is useful for human investigators. It may well be that even though the neural network can distinguish between plots for a certain task, human investigators cannot learn to do so. But by computing the TVS prior to conducting human subject experiments, we provide a more informed foundation for designing the experiments, which may make it possible to separate two equally important aspects of visual inference, namely 1) whether the visual inference task is technically possible, that is, whether the plot actually contains the necessary information; and 2) whether humans can learn to interpret the plots correctly. In current studies of visual inference using human subjects, these aspects are assessed jointly and therefore, it is difficult to reach clear conclusions about why a visual inference method may fail to produce the expected results. Especially the second aspect is notoriously difficult to study because it may depend on the specific graphical design and the specific experimental setting. We therefore believe it will be very useful to be able to address 1) separately before we need to worry about the more difficult question 2).

On a similar note, the TVS values obtained in the application for small values of n may be somewhat surprising. For instance, for data generating mechanism C, $\sigma^2 = 1$ and only $n = 5$ observations for each plot, the scatter plot – which is the best performer on that task – obtains a TVS of 0.81. We do not believe it is likely that a human investigator can achieve the same accuracy for this task. But that is entirely beside the point; stand-alone TVS values will generally not be very interesting, unless we actually want the machine learning algorithm to help us classify plots. TVS values are intended for comparing different plots, and as we have shown with the calibration scale in Figure 4, having access to a selection of TVS values for a well-known plotting tasks may help us in interpreting TVS values for new plotting tasks. Without such a scale, on the other hand, the absolute values of TVS scores will generally be difficult to make use of, except if they take the extreme values of 0.5 (random guessing) or 1 (perfect classification).

Our specific implementation suggested in Sections 3 and 4 choices may not be useful for all validation tasks, but we believe that the overall protocol can be modified to fit a lot of different needs. For instance, even though we have defined the TVS as the validation data classification accuracy, it could easily be replaced by a different performance metric, such as area under the ROC curve or F1-score, if an application calls for a different trade-off between sensitivity and specificity.

Similarly, the choice of the Inception network can easily be replaced with a different method for constructing a classification function. However, we advocate against developing a new model for each validation task, as one will then lose the opportunity to calibrate the procedure and thus the obtained TVS values will be difficult to interpret. So even though convolutional networks are notoriously known for requiring extensive application-specific network architecture engineering to obtain the best performance, we believe that our proposed 'one-size-fits-all' solution is more useful for validating and comparing plots: We do not need to obtain the best possible performances for each plot classification task, as long as the performances across different tasks can be compared. And that is exactly what the 'one-size-fits-all' approach allows us to do.

A strength of our proposed method is that it makes it possible to assess the validity of a visual inference method for binary decisions for any given within-plot sample size. In contrast, classical statistical tests rely on asymptotic properties, and therefore they often do not have well-known small sample properties. Hence, in practice, decisions are often made based on so-called "rules of thumb" that have inadequate scientific foundation. Thus, by providing a method for validating statistical decision making procedures that can be used for all sample sizes, we address a common struggle in applied statistics.

As mentioned in Section 3, in the most recent years, computer vision research has moved its focus from image classification to object detection. Object detection involves identifying *where* in an image a certain object occurs, for example by drawing a tight box around the object. It could be very interesting to follow this development in connection with visual inference as well. For instance, if models could be developed to identify *where* in a plot we need to look in order to identify misspecification, it would allow both for more informed model diagnostics based on visual inference and a promising avenue for developing learning tools that could help students in learning how to interpret diagnostic plots correctly. This topic should be addressed in future research.

Appendix A: Details about the expected impact of changing the variance

We here provide further details about the expected impact of σ^2 on the TVS. We discuss each data generating mechanism in turn.

Case A corresponds to a non-linear (exponential) regression with Gaussian errors. The variance, $sigma^2$, determines the range of the observed X values, with larger values of σ resulting in larger ranges of X. When $sigma$ is small, then the range of X, and consequently $\exp(X)$, is limited, and the curvature of the exponential function will be overshadowed by the residual error, ϵ. When $sigma$ increases, the range of $\exp(X)$ increases and the non-linear curvature will be more readily observable relative to the residual variance. Consequently, it will be easier to identify the alternative data generating mechanisms when σ increases. This will be the case for all three plots since the discrepancy between the data generating process and the model will increase with larger σ.

Case B corresponds to a linear regression analysis with non-normal errors. The ordinary least squares estimates used to obtain the linear regression model fit will in general provide unbiased predictions in this case. Since X_i and ϵ_i are independent, an increase in $sigma^2$ will translate directly to a similar increase in variance of Y. In fact, for fixed ϵ, we have that any affine transformation of the Xs will – apart from the numbers showing the scale on the x and y axes – produce the exact same visual scatter, residual, and QQ plots. And because both X and Y values are standardized prior to plotting, even the scales will be the same. Consequently, the variance should have no impact on the discrimination of the alternative data generating mechanism for case B.

Case C corresponds to the situation of a log-normal distribution with Berkson errors, that is, where the observed X can be thought of as being partly contaminated since it is not the true underlying value resulting in Y. The sum of $X + \epsilon$ will itself follow a normal distribution (with variance 1.5, 2, and 3), and while the data-generating process adds no errors to the predicted values, the increasing variance will spread out the range of Xs and will make the exponential relationship clearer. Especially so as the errors in the X variables used for fitting the linear regression model will result in asymmetric effects on the Y values: The curvature of $\exp(X + \epsilon)$ will result in relatively larger values of the outcome when $\epsilon > 0$. Thus, the overall exponential form will be kept, while the errors will have relatively smaller impact as the variance of X increases. Consequently, we expect better performance as the variance increases.

Supplementary materials

Plot collection
Data set used for training the neural networks. Organized in a nested folder structure and the lowest level folders contain image files. (.zip folder with folders containing .png image files). Available at `http://publicifsv.sund.ku.dk/~ahp/visval/visvalplots.zip`.

Replication code
R code used for the examples. Available at `https://github.com/annennenne/visval`.

Acknowledgements

This work was supported by the Independent Research Fund Denmark under Grant 8020-00031B.

References

[1] Jenny J Chen, Natala J Menezes, Adam D Bradley, and T North. Opportunities for crowdsourcing research on amazon mechanical turk. *Interfaces*, 5(3):1, 2011.

[2] Susan VanderPlas and Heike Hofmann. Clusters beat trend!? testing feature hierarchy in statistical graphics. *Journal of Computational and Graphical Statistics*, 26(2):231–242, 2017.

[3] Alex Kale, Matthew Kay, and Jessica Hullman. Visual reasoning strategies and satisficing: How uncertainty visualization design impacts effect size judgments and decisions. *arXiv preprint arXiv:2007.14516*, 2020.

[4] Jeffrey Heer and Michael Bostock. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 203–212, 2010.

[5] Jessica Hullman, Paul Resnick, and Eytan Adar. Hypothetical outcome plots outperform error bars and violin plots for inferences about reliability of variable ordering. *PloS one*, 10(11):e0142444, 2015.

[6] Adam Loy, Heike Hofmann, and Dianne Cook. Model choice and diagnostics for linear mixed-effects models using statistics on street corners. *Journal of Computational and Graphical Statistics*, 26(3):478–492, 2017.

[7] Mahbubul Majumder, Heike Hofmann, and Dianne Cook. Validation of visual statistical inference, applied to linear models. *Journal of the American Statistical Association*, 108(503):942–956, 2013.

[8] Joel Ross, Andrew Zaldivar, Lilly Irani, and Bill Tomlinson. Who are the turkers? worker demographics in amazon mechanical turk. *Department of Informatics, University of California, Irvine, USA, Tech. Rep*, 2009.

[9] Bryan FJ Manly. *Randomization, bootstrap and Monte Carlo methods in biology*, volume 70. CRC press, 2006.

[10] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[11] Bradley Efron and Trevor Hastie. *Computer age statistical inference*, volume 5. Cambridge University Press, 2016.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[14] Julian J Faraway. *Linear models with R.* CRC press, 2014.

[15] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2020.

[16] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York, 2016.

[17] Microsoft and Steve Weston. *foreach: Provides Foreach Looping Construct*, 2020. R package version 1.5.0.

[18] Revolution Analytics and Steve Weston. *doMC: Foreach Parallel Adaptor for 'parallel'*, 2019. R package version 1.3.6.

[19] François Chollet, JJ Allaire, et al. R interface to keras. `https://github.com/rstudio/keras`, 2017.

[20] Andreas Buja, Dianne Cook, Heike Hofmann, Michael Lawrence, Eun-Kyung Lee, Deborah F Swayne, and Hadley Wickham. Statistical inference for exploratory data analysis and model diagnostics. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1906):4361–4383, 2009.

[21] Claus Thorn Ekstrøm. Teaching 'instant experience'with graphical model validation techniques. *Teaching Statistics*, 36(1):23–26, 2014.