DRAFT PAPER (INCOMPLETE)

Appendix: Automated assessment of residual plots with computer vision models

Weihao Li[a], Dianne Cook[a], Emi Tanaka[b,c], Susan VanderPlas[d], Klaus Ackermann[a]

[a]Department of Econometrics and Business Statistics, Monash University, Clayton, VIC, Australia; [b]Biological Data Science Institute, Australian National University, Acton, ACT, Australia; [c]Research School of Finance, Actuarial Studies and Statistics, Australian National University, Acton, ACT, Australia; [d]Department of Statistics, University of Nebraska, Lincoln, Nebraska, USA

ARTICLE HISTORY

Compiled May 7, 2024

ABSTRACT

TBD.

KEYWORDS

TBD

1. Additional details about data generation

1.1. *Computation of scagnostics*

In Section **??**, we mentioned that scagnostics consist of a set of manually designed visual feature extraction functions. While our computer vision model will learn its own feature extraction function during training, leveraging additional information from scagnostics can enhance the model's predictive accuracy.

CONTACT Weihao Li. Email: `weihao.li@monash.edu`, Dianne Cook. Email: `dicook@monash.edu`, Emi Tanaka. Email: `emi.tanaka@anu.edu.au`, Susan VanderPlas. Email: `susan.vanderplas@unl.edu`, Klaus Ackermann. Email: `Klaus.Ackermann@monash.edu`

For each generated residual plot, we computed four scagnostics – "Monotonic," "Sparse," "Splines," and "Striped" – using the `cassowaryr` R package (Mason et al. 2022). These computed measures, along with the number of observations from the fitted model, were provided as the second input for the computer vision model. Although other scagnostics are informative, they are currently unavailable due to a fatal bug in the compiled C program of the `interp` R package (Gebhardt, Bivand, and Sinclair 2023), which may unpredictably crash the process. For reproducibility, we excluded these scagnostics from the training data.

2. Neural Network Layers Used in the Study

This study employs seven types of neural network layers, all of which are standard components frequently found in modern deep learning models. These layers are well-documented in textbooks like Goodfellow, Bengio, and Courville (2016), which offer thorough explanations and mathematical insights. In this section, we will offer a concise overview of these layers, drawing primarily from the insights provided in Goodfellow, Bengio, and Courville (2016).

2.1. *Dense Layer*

The Dense layer, also known as the fully-connected layer, is the fundamental unit in neural networks. It conducts a matrix multiplication operation between the input matrix $\boldsymbol{X}$ and a weight matrix $\boldsymbol{W}$ to generate the output matrix $\boldsymbol{O}$, which can be written as

$$\boldsymbol{O} = \boldsymbol{X}\boldsymbol{W} + b,$$

where b is the intercept.

2.2. *ReLu Layer*

The ReLU layer, short for rectified linear unit, is an element-wise non-linear function introduced by Nair and Hinton (2010). It sets the output elements to zero if the corresponding input element is negative; otherwise, it retains the original input. Mathematically, it can be expressed as:

$$\boldsymbol{O}(i,j) = max\{0, \boldsymbol{X}(i,j)\},$$

where $\boldsymbol{O}(i,j)$ is the ith row and jth column entry of matrix $\boldsymbol{O}$, and $\boldsymbol{X}(i,j)$ is the ith row and jth column entry of matrix $\boldsymbol{X}$.

2.3. *Convolutaional Layer*

In Dense layers, matrix multiplication leads to each output unit interacting with every input unit, whereas convolutional layers operate differently with sparse interactions. Here, an output unit in a convolutional layer is connected solely to a subset of input units, and the weight is shared across all input units. Achieving this involves using a kernel, typically a small square matrix, to conduct matrix multiplication across all input units. Precisely, this concept can be formulated as:

$$O(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n),$$

where m and n are the row and columns indices of the kernel K.

If there are multiple kernels used in one covolutional layer, then each kernel will have its own weights and the output will be a three-dimensional tensor, where the length of the third channel is the number of kernels.

2.4. *Pooling Layer*

A pooling layer substitutes the input at a specific location with a summary statistic derived from nearby input units. Typically, there are two types of pooling layers: max pooling and average pooling. Max pooling computes the maximum value within a rectangular neighborhood, while average pooling calculates their average. Pooling layers helps making the representation approximately invariant to minor translations of the input. The output matrix of a pooling layer is approximately s times smaller than the input matrix, where s represents the length of the rectangular area. This can be formulated as:

$$O(i, j) = \max_{m,n} I(si + m, sj + n).$$

2.5. *Global Pooling Layer*

A global pooling layer condenses an input matrix into a scalar value by either extracting the maximum or computing the average across all elements. This layer acts as a crucial link between the convolutional structure and the subsequent dense layers in a neural network architecture. When convolutional layers utilize multiple kernels, the output becomes a three-dimensional tensor with numerous channels. In this scenario, the global pooling layer treats each channel individually, much like distinct features in a conventional classifier. This approach facilitates the extraction of essential features from complex data representations, enhancing the network's ability to learn meaningful patterns.

2.6. *Batch Normalization Layer*

2.7. *Dropout Layer*

References

Gebhardt, Albrecht, Roger Bivand, and David Sinclair. 2023. *interp: Interpolation Methods.* R package version 1.1-5, `https://CRAN.R-project.org/package=interp`.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Mason, Harriet, Stuart Lee, Ursula Laa, and Dianne Cook. 2022. *cassowaryr: Compute Scagnostics on Pairs of Numeric Variables in a Data Set*. R package version 2.0.0, `https://CRAN.R-project.org/package=cassowary`.

Nair, Vinod, and Geoffrey E Hinton. 2010. "Rectified linear units improve restricted boltzmann machines." In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.