

Appendix to “Automated Assessment of Residual Plots with Computer Vision Models”

Weihaio Li^{a,b}, Dianne Cook^a, Emi Tanaka^{a,b,c}, Susan VanderPlas^d, Klaus Ackermann^a

^aDepartment of Econometrics and Business Statistics, Monash University, Clayton, VIC, Australia; ^bBiological Data Science Institute, Australian National University, Acton, ACT, Australia; ^cResearch School of Finance, Actuarial Studies and Statistics, Australian National University, Acton, ACT, Australia; ^dDepartment of Statistics, University of Nebraska, Lincoln, Nebraska, USA

ARTICLE HISTORY

Compiled June 12, 2025

ABSTRACT

Plotting the residuals is a recommended procedure to diagnose deviations from linear model assumptions, such as non-linearity, heteroscedasticity, and non-normality. The presence of structure in residual plots can be tested using the lineup protocol to do visual inference. There are a variety of conventional residual tests, but the lineup protocol, used as a statistical test, performs better for diagnostic purposes because it is less sensitive and applies more broadly to different types of departures. However, the lineup protocol relies on human judgment which limits its scalability. This work presents a solution by providing a computer vision model to automate the assessment of residual plots. It is trained to predict a distance measure that quantifies the disparity between the residual distribution of a fitted classical normal linear regression model and the reference distribution, based on Kullback-Leibler divergence. From extensive simulation studies, the computer vision model exhibits lower sensitivity than conventional tests but higher sensitivity than human visual tests. It is slightly less effective on non-linearity patterns. Several examples from classical papers and contemporary data illustrate the new procedures, highlighting its usefulness in automating the diagnostic process and supplementing existing methods.

KEYWORDS

statistical graphics; data visualization; visual inference; computer vision; machine learning; hypothesis testing; regression analysis; cognitive perception; simulation; practical significance

1. Data Generation

1.1. *Simulation Scheme*

While observational data is frequently employed in training models for real-world applications, the data generating process of observational data often remains unknown, making computation for our target variable D unattainable. Consequently, the computer vision models developed in this study were trained using synthetic data, including 80,000 training images and 8,000 test images. This approach provided us with precise label annotations. Additionally, it ensured a large and diverse training dataset, as we had control over the data generating process, and the simulation of the training data was relatively cost-effective.

We have incorporated three types of residual departures of linear regression model in the training data, including non-linearity, heteroskedasticity and non-normality. All three departures can be summarized by the data generating process formulated as

$$\mathbf{y} = \mathbf{1}_n + \mathbf{x}_1 + \beta_1 \mathbf{x}_2 + \beta_2 (\mathbf{z} + \beta_1 \mathbf{w}) + \mathbf{k} \odot \boldsymbol{\varepsilon}, \quad (1)$$

$$\mathbf{z} = \text{He}_j(g(\mathbf{x}_1, 2)), \quad (2)$$

$$\mathbf{w} = \text{He}_j(g(\mathbf{x}_2, 2)), \quad (3)$$

$$\mathbf{k} = [\mathbf{1}_n + b(2 - |a|)(\mathbf{x}_1 + \beta_1 \mathbf{x}_2 - a\mathbf{1}_n)^{\odot 2}]^{\odot 1/2}, \quad (4)$$

where \mathbf{y} , \mathbf{x}_1 , \mathbf{x}_2 , \mathbf{z} , \mathbf{w} , \mathbf{k} and $\boldsymbol{\varepsilon}$ are vectors of size n , $\mathbf{1}_n$ is a vector of ones of size n , \mathbf{x}_1 and \mathbf{x}_2 are two independent predictors, $\text{He}_j(\cdot)$ is the j th-order probabilist's Hermite polynomials (Hermite 1864), $(\cdot)^{\odot 2}$ and $(\cdot)^{\odot 1/2}$ are Hadamard square and square root, \odot is the Hadamard product, and $g(\mathbf{x}, k)$ is a scaling function to enforce the support of the random vector to be $[-k, k]^n$ defined as

Table 1. Factors used in the data generating process for synthetic data simulation. Factor j and a controls the non-linearity shape and the heteroskedasticity shape respectively. Factor b , σ_ε and n control the signal strength. Factor dist_ε , dist_{x_1} and dist_{x_2} specifies the distribution of ε , X_1 and X_2 respectively.

Factor	Domain
j	$\{2, 3, \dots, 18\}$
a	$[-1, 1]$
b	$[0, 100]$
β_1	$\{0, 1\}$
β_2	$\{0, 1\}$
dist_ε	$\{\text{discrete, uniform, normal, lognormal}\}$
dist_{x_1}	$\{\text{discrete, uniform, normal, lognormal}\}$
dist_{x_2}	$\{\text{discrete, uniform, normal, lognormal}\}$
σ_ε	$[0.0625, 9]$
σ_{X1}	$[0.3, 0.6]$
σ_{X2}	$[0.3, 0.6]$
n	$[50, 500]$

$$g(\mathbf{x}, k) = 2k \cdot \frac{\mathbf{x} - x_{\min} \mathbf{1}_n}{x_{\max} - x_{\min}} - k \mathbf{1}_n, \text{ for } k > 0,$$

where $x_{\min} = \min_{i \in \{1, \dots, n\}} x_i$, $x_{\max} = \max_{i \in \{1, \dots, n\}} x_i$ and x_i is the i -th entry of \mathbf{x} .

The residuals and fitted values of the fitted model were obtained by regressing \mathbf{y} on \mathbf{x}_1 . If $\beta_1 \neq 0$, \mathbf{x}_2 was also included in the design matrix. This data generation process was adapted from Li et al. (2024), where it was utilized to simulate residual plots exhibiting non-linearity and heteroskedasticity visual patterns for human subject experiments. A summary of the factors utilized in Equation 1 is provided in Table 1.

In Equation 1, \mathbf{z} and \mathbf{w} represent higher-order terms of \mathbf{x}_1 and \mathbf{x}_2 , respectively. If $\beta_2 \neq 0$, the regression model will encounter non-linearity issues. Parameter j serves as a shape parameter that controls the number of tuning points in the non-linear pattern. Typically, higher values of j lead to an increase in the number of tuning points, as illustrated in Figure 1.

Additionally, scaling factor k directly affects the error distribution and it is correlated with \mathbf{x}_1 and \mathbf{x}_2 . If $b \neq 0$ and $\varepsilon \sim N(\mathbf{0}_n, \sigma^2 \mathbf{I}_n)$, the constant variance assumption will be violated. Parameter a is a shape parameter controlling the location of the

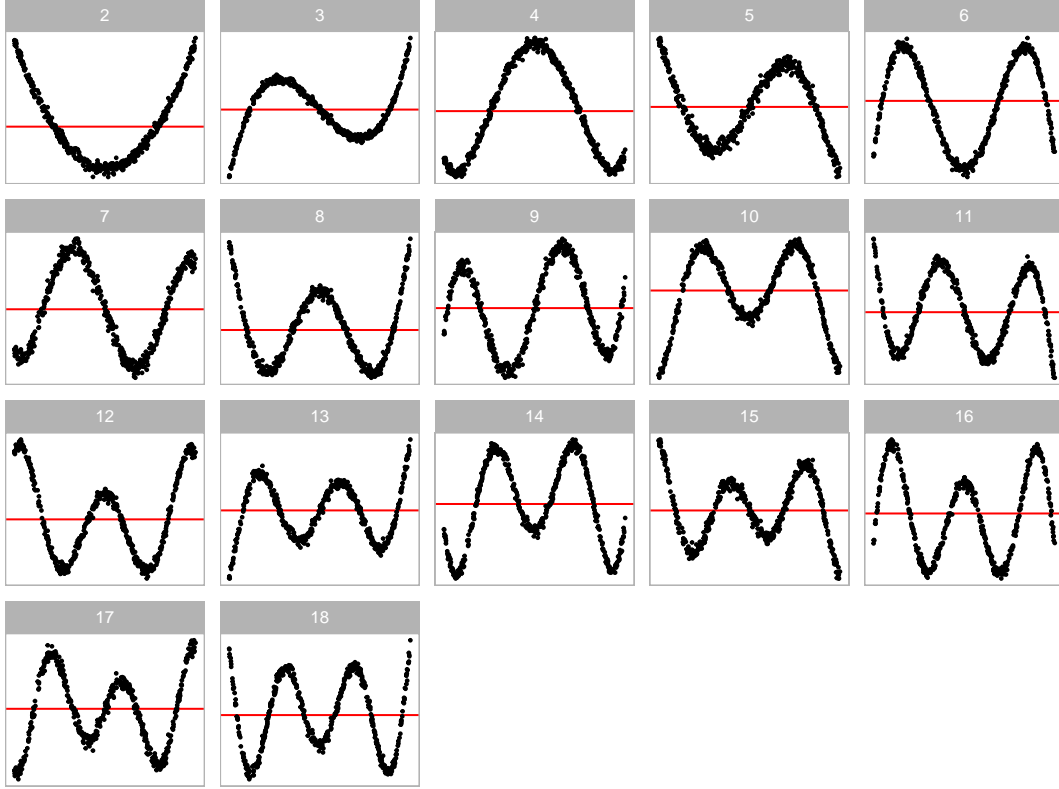


Figure 1. Non-linearity forms generated for the synthetic data simulation. The 17 shapes are generated by varying the order of polynomial given by j in $He_j(\cdot)$.

smallest variance in a residual plot as shown in Figure 2.

Non-normality violations arise from specifying a non-normal distribution for ϵ . In the synthetic data simulation, four distinct error distributions are considered, including discrete, uniform, normal, and lognormal distributions, as presented in Figure 3. Each distribution imparts unique characteristics in the residual plot. The discrete error distribution introduces discrete clusters in residuals, while the lognormal distribution typically yields outliers. Uniform error distribution may result in residuals filling the entire space of the residual plot. All of these distributions exhibit visual distinctions from the normal error distribution.

Equation 1 accommodates the incorporation of the second predictor \mathbf{x}_2 . Introducing it into the data generation process by setting $\beta_1 = 1$ significantly enhances the complexity of the shapes, as illustrated in Figure 4. In comparison to Figure 1, Figure 4 demonstrates that the non-linear shape resembles a surface rather than a single curve. This augmentation can facilitate the computer vision model in learning visual

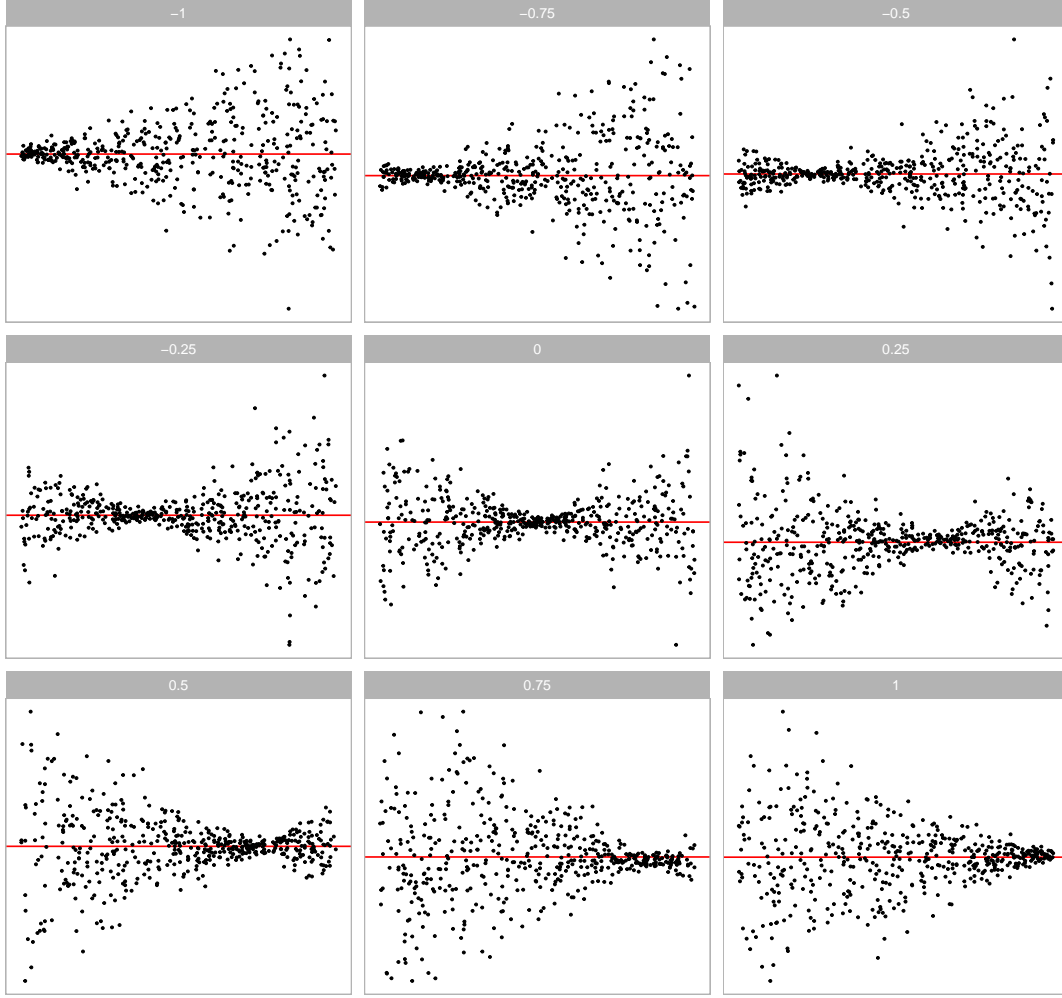


Figure 2. Heteroskedasticity forms generated for the synthetic data simulation. Different shapes are controlled by the continuous factor a between -1 and 1. For $a = -1$, the residual plot exhibits a "left-triangle" shape. And for $a = 1$, the residual plot exhibits a "right-triangle" shape.

patterns from residual plots of the multiple linear regression model.

In real-world analysis, it is not uncommon to encounter instances where multiple model violations coexist. In such cases, the residual plots often exhibit a mixed pattern of visual anomalies corresponding to different types of model violations. Figure 5 and Figure 6 show the visual patterns of models with multiple model violations.

The predictors, \mathbf{x}_1 and \mathbf{x}_2 , are randomly generated from four distinct distributions, including $U(-1, 1)$ (uniform), $N(0, 0.3^2)$ (normal), $\text{lognormal}(0, 0.6^2)/3$ (skewed) and $U\{-1, 1\}$ (discrete uniform).

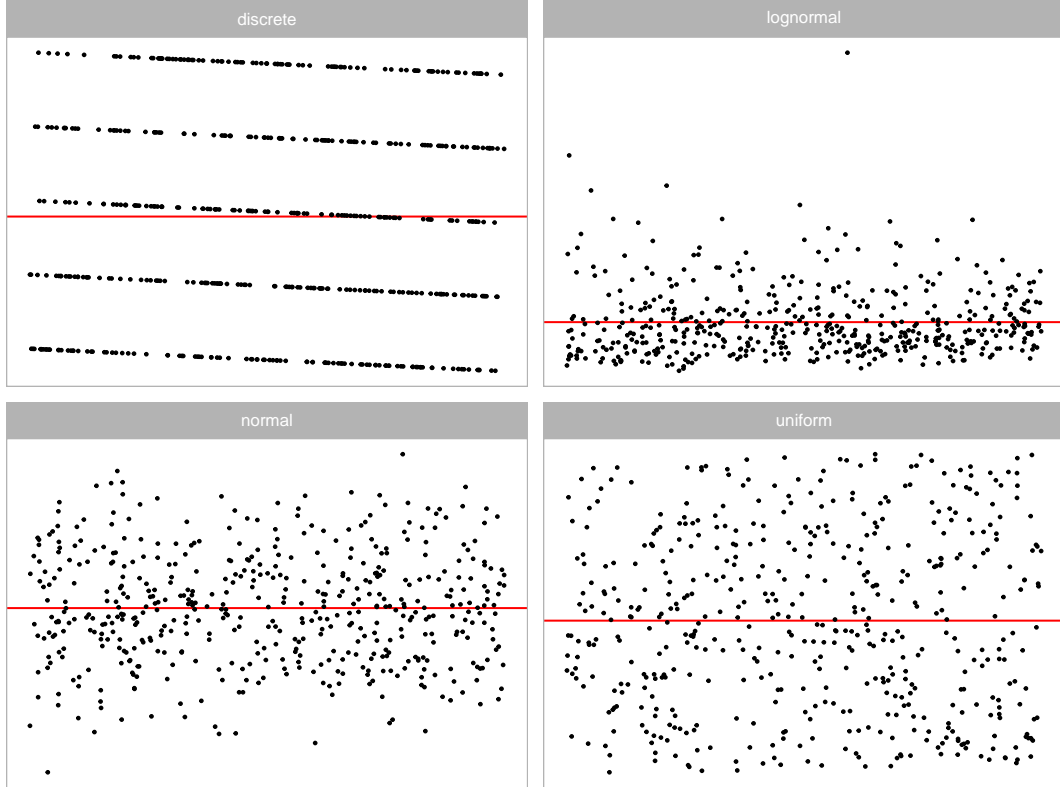


Figure 3. Non-normality forms generated for the synthetic data simulation. Four different error distributions including discrete, lognormal, normal and uniform are considered.

1.2. *Balanced Dataset*

To train a robust computer vision model, we deliberately controlled the distribution of the target variable D in the training data. We ensured that it followed a uniform distribution between 0 and 7. This was achieved by organizing 50 buckets, each exclusively accepting training samples with D falling within the range $[7(i-1)/49, 7i/49)$ for $i < 50$, where i represents the index of the i -th bucket. For the 50-th bucket, any training samples with $D \geq 7$ were accepted.

With 80,000 training images prepared, each bucket accommodated a maximum of $80000/50 = 1600$ training samples. The simulator iteratively sampled parameter values from the parameter space, generated residuals and fitted values using the data generation process, computed the distance, and checked if the sample fitted within the corresponding bucket. This process continued until all buckets were filled.

Similarly, we adopted the same methodology to prepare 8,000 test images for performance evaluation and model diagnostics.

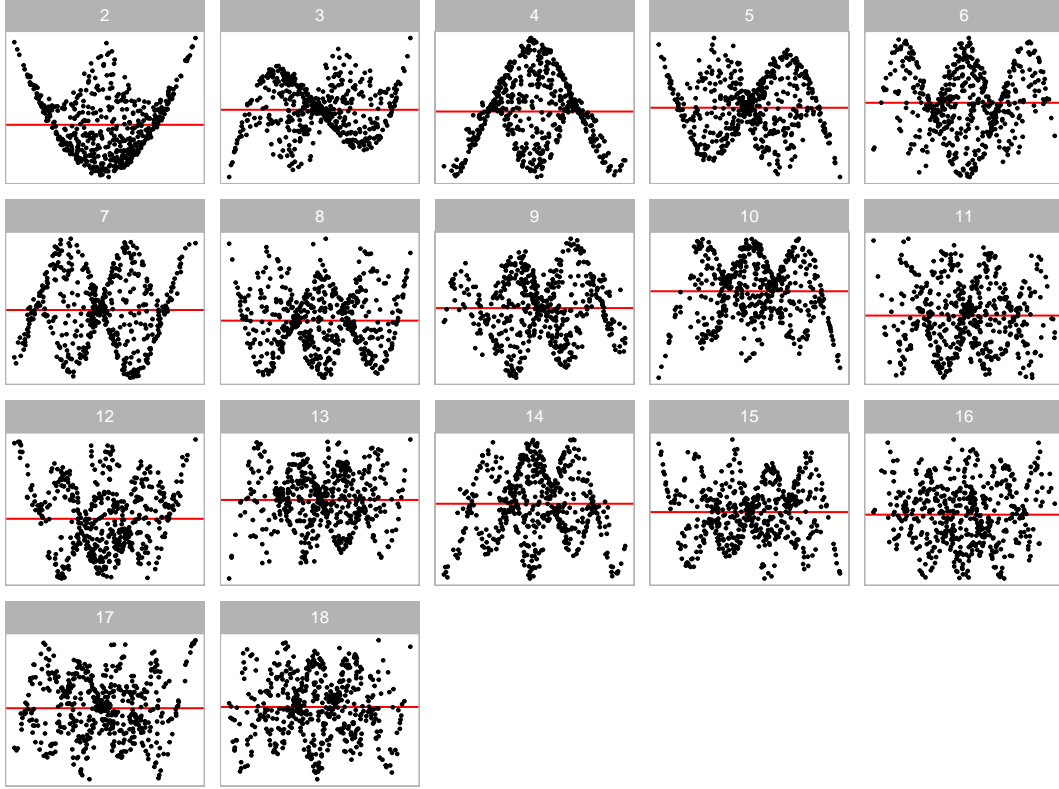


Figure 4. Residual plots of multiple linear regression models with non-linearity issues. The 17 shapes are generated by varying the order of polynomial given by j in $He_j(\cdot)$. A second predictor \mathbf{x}_2 is introduced to the regression model to create complex shapes.

2. Model Training

To achieve a near-optimal deep learning model, hyperparameters like the learning rate often need to be fine-tuned using a tuner. In our study, we utilized the Bayesian optimization tuner from the `KerasTuner` Python library (O’Malley et al. 2019) for this purpose. A comprehensive list of hyperparameters is provided in Table 2.

The number of base filters determines the number of filters for the first 2D convolutional layer. In the VGG16 architecture, the number of filters for the 2D convolutional layer in a block is typically twice the number in the previous block, except for the last block, which maintains the same number of convolution filters as the previous one. This hyperparameter aids in controlling the complexity of the computer vision model. A higher number of base filters results in more trainable parameters. Likewise, the number of units for the fully-connected layer determines the complexity of the final prediction block. Increasing the number of units enhances model complexity, resulting

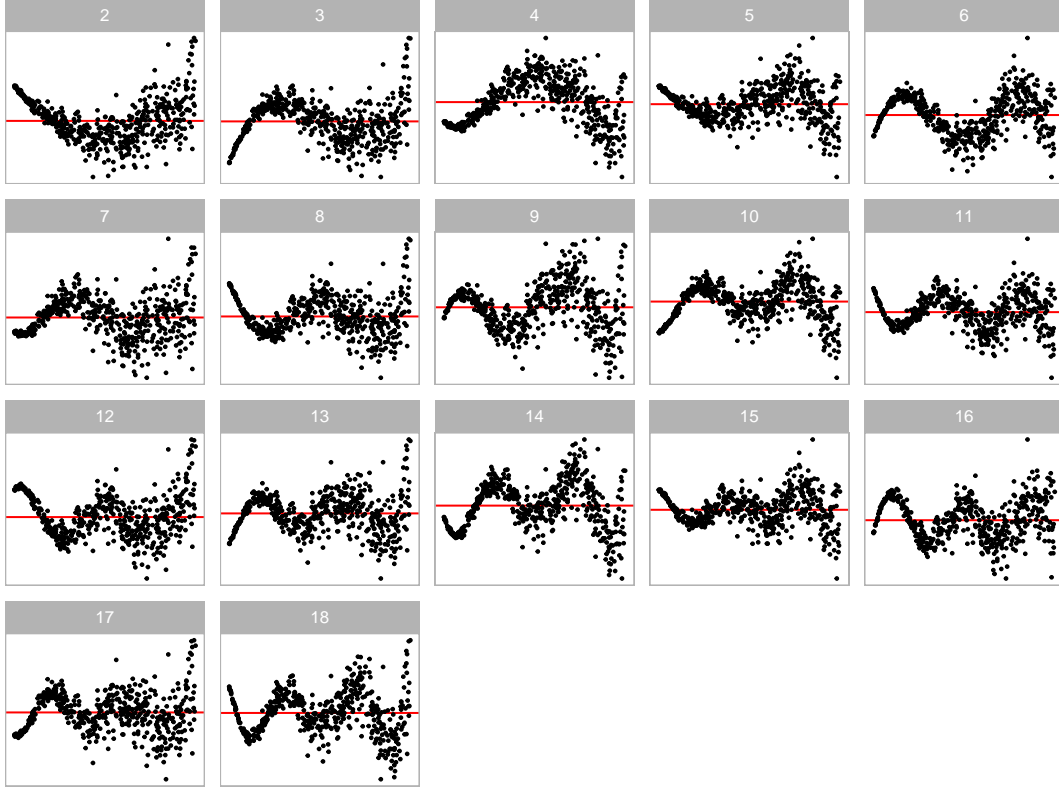


Figure 5. Residual plots of models violating both the non-linearity and the heteroskedasticity assumptions. The 17 shapes are generated by varying the order of polynomial given by j in $He_j(\cdot)$, and the "left-triangle" shape is introduced by setting $a = -1$.

in more trainable parameters.

The dropout rate and batch normalization are flexible hyperparameters that work in conjunction with other parameters to facilitate smooth training. A higher dropout rate is necessary when the model tends to overfit the training dataset by learning too much noise (Srivastava et al. 2014). Conversely, a lower dropout rate is preferred when the model is complex and challenging to converge. Batch normalization, on the other hand, addresses the internal covariate shift problem arising from the randomness in weight initialization and input data (Goodfellow, Bengio, and Courville 2016). It helps stabilize and accelerate the training process by normalizing the activations of each layer.

Additionally, incorporating additional inputs such as scagnostics and the number of observations can potentially enhance prediction accuracy. Therefore, we allow the tuner to determine whether these inputs were necessary for optimal model performance.

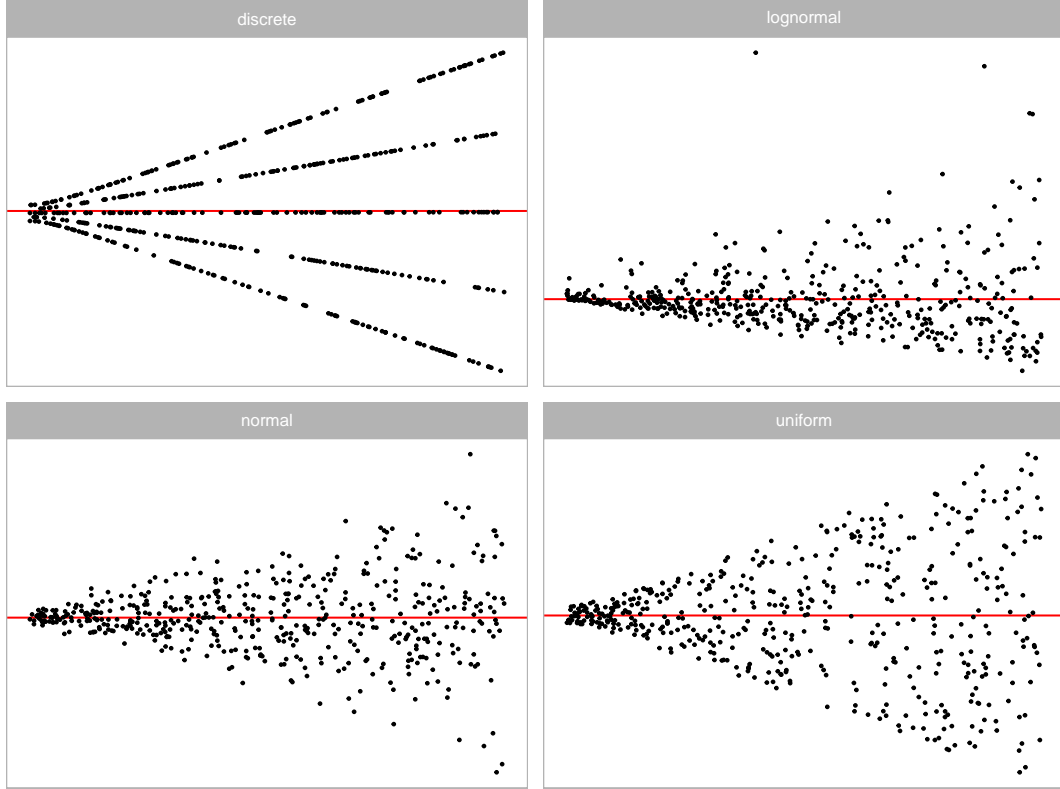


Figure 6. Residual plots of models violating both the non-normality and the heteroskedasticity assumptions. The four shapes are generated by using four different error distributions including discrete, lognormal, normal and uniform, and the "left-triangle" shape is introduced by setting $a = -1$.

The learning rate is a crucial hyperparameter, as it dictates the step size of the optimization algorithm. A high learning rate can help the model avoid local minima but risks overshooting and missing the global minimum. Conversely, a low learning rate smoothens the training process but makes the convergence time longer and increases the likelihood of getting trapped in local minima.

Our model was trained on the MASSIVE M3 high-performance computing platform (Goscinski et al. 2014), using TensorFlow (Abadi et al. 2016) and Keras (Chollet et al. 2015). During training, 80% of the training data was utilized for actual training, while the remaining 20% was used as validation data. The Bayesian optimization tuner conducted 100 trials to identify the best hyperparameter values based on validation root mean square error. The tuner then restored the best epoch of the best model from the trials. Additionally, we applied early stopping, terminating the training process if the validation root mean square error fails to improve for 50 epochs. The maximum allowed epochs was set at 2,000, although no models reached this threshold.

Table 2. Name of hyperparameters and their corresponding domain for the computer vision model.

Hyperparameter	Domain
Number of base filters	{4, 8, 16, 32, 64}
Dropout rate for convolutional blocks	[0.1, 0.6]
Batch normalization for convolutional blocks	{false, true}
Type of global pooling	{max, average}
Ignore additional inputs	{false, true}
Number of units for the fully-connected layer	{128, 256, 512, 1024, 2048}
Batch normalization for the fully-connected layer	{false, true}
Dropout rate for the fully-connected layer	[0.1, 0.6]
Learning rate	$[10^{-8}, 10^{-1}]$

Table 3. Hyperparameters values for the optimized computer vision models with different input sizes.

Hyperparameter	32×32	64×64	128×128
Number of base filters	32	64	64
Dropout rate for convolutional blocks	0.4	0.3	0.4
Batch normalization for convolutional blocks	true	true	true
Type of global pooling	max	average	average
Ignore additional inputs	false	false	false
Number of units for the fully-connected layer	256	256	256
Batch normalization for the fully-connected layer	false	true	true
Dropout rate for the fully-connected layer	0.2	0.4	0.1
Learning rate	0.0003	0.0006	0.0052

Based on the tuning process described above, the optimized hyperparameter values are presented in Table 3. It was observable that a minimum of 32 base filters was necessary, with the preferable choice being 64 base filters for both the 64×64 and 128×128 models, mirroring the original VGG16 architecture. The optimized dropout rate for convolutional blocks hovered around 0.4, and incorporating batch normalization for convolutional blocks proved beneficial for performance.

All optimized models chose to retain the additional inputs, contributing to the reduction of validation error. The number of units required for the fully-connected layer was 256, a relatively modest number compared to the VGG16 classifier, suggesting that the problem at hand was less complex. The optimized learning rates were higher for models with higher resolution input, likely because models with more parameters are more prone to getting stuck in local minima, requiring a higher learning rate.

References

- Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, et al. 2016. “Tensorflow: Large-scale machine learning on heterogeneous distributed systems.” *arXiv preprint arXiv:1603.04467* .
- Chollet, François, et al. 2015. “Keras.” <https://keras.io>.
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.
- Goscinski, Wojtek J, Paul McIntosh, Ulrich Felzmann, Anton Maksimenko, Christopher J Hall, Timur Gureyev, Darren Thompson, et al. 2014. “The multi-modal Australian ScienceS Imaging and Visualization Environment (MASSIVE) high performance computing infrastructure: applications in neuroscience and neuroinformatics research.” *Frontiers in Neuroinformatics* 8: 30.
- Hermite, M. 1864. *Sur un nouveau développement en série des fonctions*. Imprimerie de Gauthier-Villars.
- Li, Weihao, Dianne Cook, Emi Tanaka, and Susan VanderPlas. 2024. “A plot is worth a thousand tests: Assessing residual diagnostics with the lineup protocol.” *Journal of Computational and Graphical Statistics* 1–19.
- O’Malley, Tom, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. 2019. “Keras Tuner.” <https://github.com/keras-team/keras-tuner>.
- Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. “Dropout: a simple way to prevent neural networks from overfitting.” *The journal of machine learning research* 15 (1): 1929–1958.