AMC JOURNAL (DHANGADHI)
Year 9; July, 2026; Vol. 8; Issue 2; PP. (29-38) DOI: https://doi.org/10.3126/amcjd.v8i2.98373

A Lightweight Neural Inference Engine for Real-Time Parameter Estimation of the Modified Inverse Generalized Exponential Distribution

Sagun Pahari[1a] , Lal Babu Sah Telee[2b*], Durga Pokharel[3c]
[1]Department of Computer Engineering, Everest Engineering College, Pokhara University, Nepal
[2]Department of Management Science, Nepal Commerce Campus, Tribhuvan University, Nepal
[3]Department of IT, Herald College, Naxal, Kathmandu, Nepal
[a]Email: sagunpahari666@gmail.com, ORCID: 0009-0003-1243-587X
[b]Email: lalbabu3131@gmail.com, ORCID: 0009-0009-0994-6445
[c]Email: pokhareldurga88@gmail.com
*Corresponding Author: lalbabu3131@gmail.com

Article History:
Received: March 20, 2026
Revised: June 10, 2026
Accepted: July 2, 2026

Abstract

The Modified Inverse Generalized Exponential (MIGE) distribution is a very useful probability distribution used in lifetime data analysis. Classical parameter estimation methods (maximum likelihood, least squares, Cramer-von-Mises) are iterative, slow (~200 ms per sample) and limiting real-time applications. This study presents a lightweight neural network that learns the direct mapping from sample quantiles to MIGE parameters. The network has only 32 hidden neurons and 419 trainable weights which is small enough to run on edge devices. Training it on 20,000 synthetic datasets (with input/output scaling and hyperparameter tuning), achievements of the neural estimator inference times of 0.0001 seconds per sample (size 100) which is 2,000× faster than maximum likelihood when used on a standard Intel i5 CPU. The neural estimates (α = 23.91, β = 0.2366, λ = 15.90) are nearly close to the maximum likelihood estimates (α = 30.78, β = 0.1942, λ=14.83), having a Kolmogorov–Smirnov statistic of 0.0789 (below the 5% critical value of 0.171, which indicates an acceptable fit) on the benchmark carbon fiber strength dataset. Bootstrap standard errors for α=3.83, β=0.0694, λ=2.93, reveals uncertainty quantification. Over 100 test sets of simulation study show a speed-accuracy trade-off has negligible bias (α = -1.79, β = -0.0059, λ = -0.097), and the neural estimator perform better than method of moments and quantile matching in accuracy during retaining speed. This study discusses how a computer engineer could further decrease MSE by implementing deeper networks (using TensorFlow Lite), deploying on FPGA/edge TPU, or quantizing weights. All codes used are open source. This study helps in bridging statistical reliability modeling with efficient embedded inference.

Keywords: MIGE distribution, carbon fiber data, edge computing, neural inference, real-time estimation

Introduction

In recent period, statisticians have developed various probability distributions to model lifetime data having non-monotonic hazard rates. The Modified Inverse Generalized Exponential (MIGE) distribution (Telee & Kumar, 2023) is one such flexible model. Its parameters are typically estimated by iterative maximum likelihood (MLE), which is accurate but computationally heavy (≈0.2 seconds per sample). Computer engineers designing real-time monitoring systems (e.g., predictive maintenance, quality control in carbon fiber production) need parameter estimates in milliseconds—a speed that classical MLE cannot provide.

This study introduces a lightweight neural parameter estimator which trades a small amount of accuracy for dramatic speed. The network is quite small enough which can run on resource-constrained devices (e.g., Raspberry Pi, industrial PLC). This may be the first application of a lightweight neural network to real-time parameter estimation for the MIGE distribution, where specific focus is on edge-deployable inference. This study is a proof of concept which bridges statistical reliability modeling with efficient embedded inference.

The field of lifetime data processing has proved substantial growth in flexible probability models. Engineering domain has provided significant strides in quick, inference without likelihood using neural networks. This review is basically focused on three key areas: First, generalizations of the exponential and inverted exponential distribution second, Inference based simulation and neural parameter estimation, and third, real-time inference on edge devices.

Generalizations of Exponential and Inverted Exponential Distributions

Due to memoryless property and constant hazard rate, the exponential distribution, serves as a foundational model in both reliability and survival analysis. But in real life data, more often exhibit non-constant hazard shapes with increasing, decreasing, unimodal, or bathtub-shaped. These non-constant hazard rates are merely captured by the classical exponential model. This limitation has motivated statistician for numerous extensions of the classical exponential distribution.

Generalized Inverse Exponential (GIE) distribution is a prominent extension. Mahmoud et al. (2021) studied parameter estimation for the GIE distribution under progressive Type-I censoring. Later, statistical inference for the GIE distribution was explored under joint progressively Type-II censored data (Zhang et al., 2024). A three-parameter Modified Inverse Generalized Exponential (MIGE) distribution was given by Telee and Kumar (2023) more recently by extending the inverse generalized exponential model. MIGE uses an additional shape parameter which was shown to provide an excellent fit to the carbon fiber strength dataset, and outperforming several competing probability models. Despite of these advances in custom model formulation, majority of these approaches rely on classical iterative estimation methods (MLE, LSE, CVM). These methods are computationally very intensive and unsuitable in case of real-time applications. These limitations create a clear opportunity for neural network-based alternatives for the estimation.

Simulation-Based Inference and Neural Parameter Estimation

Simulation based inference (SBI) provides a powerful alternative when likelihood functions are computationally prohibitive or intractable, Cranmer, Brehmer, and Louppe (2020) introduced a detailed review of SBI techniques. Neural methods for amortized parameter inference have emerged as a particularly promising direction under the SBI framework. Recent progress in point estimation, approximate Bayesian inference as well as automatic construction of summary statistics using neural networks was reviewed by Zammit Mangion et al. (2024).

A neural network that maps data to a point summary of the posterior distribution is notable development as neural Bayes estimator. Sainsbury-Dale, Zammit-Mangion, and Huser (2023) showed that neural Bayes estimators are likelihood-free, and amenable as well as faster to rapid bootstrap-based uncertainty quantification. Such estimators are the user-friendly development facilitated by the NeuralEstimator R package (Sainsbury-Dale, 2025).

Although having these advances, most SBI research mainly focus on complex scientific simulators. The use of neural estimation method to clearly-defined parametric distributions

such as the MIGE model remains largely unexplored. This gives an opportunity to bridge statistical modeling and efficient inference.

Real-Time Inference on Edge Devices

Computer engineers must have the ability to perform parameter estimation in case of real time on resource-constrained devices. Edge inference enables real-time processing with low latency reducing dependency on network connectivity (Nguyen, 2024). Implementation of specific hardware have showed the feasibility of extremely low-power neural inference Eciton is a very low-power recurrent neural network accelerator for time series data which achieve real-time inference with a power consumption of only 17 mW on a low-cost Lattice iCE40 UP5K FPGA given by Chen et al. (2024).

Lightweight neural architectures, quantization, and specialized hardware accelerators provides a simple neural parameter estimator for a statistical distribution like MIGE can be deployed in real-time monitoring systems which are the advances of edge AI. Despite of useful intersection of statistical lifetime modeling with edge-deployable neural inference, little attention is given to it.

The MIGE Distribution

The cumulative distribution function (CDF) of MIGE distribution is

$$F(x;\alpha,\beta,\lambda)=1-\left[1-\exp\left(-\frac{\lambda}{x}e^{-\beta x}\right)\right]^{\alpha} \quad ;\ (\alpha,\beta,\lambda)>0, x>0 \qquad (1)$$

The PDF of MIGE distribution is,

$$f(x;\alpha,\beta,\lambda)=\frac{\alpha\lambda}{x^2}(1+\beta x)\exp\left(-\beta x-\frac{\lambda}{x}e^{-\beta x}\right)\left[1-\exp\left(-\frac{\lambda}{x}e^{-\beta x}\right)\right]^{\alpha-1} \qquad (2)$$

Three parameters α, β, and λ control shape, scale, and tail heaviness of the distribution. Although, classical MLE is accurate but it is slow so we propose a neural alternative for estimation. **Since** *the quantile function of the MIGE distribution does not have a closed form so we evaluate it numerically using root-finding on the CDF which is done once during training data generation.*

Material and Method

Neural Estimator: Design for Speed and Low Footprint

Input Features: We computed nine quantiles taking probabilities 0.1, 0.2, …, 0.9 from a sample of size m=100, Preliminary experiments are performed taking 5th, 9th, and 15th quantiles providing 9th as the best balance between input dimension and estimation accuracy (based on validation MSE), which reduced the input dimension to 9, keeping the network extremely small.

Architecture

A single hidden layer multilayer perceptron (MLP):

Input layer: 9 neurons

Hidden layer: 32 neurons, ReLU activation

Output layer: 3 neurons for log α, log β, and log λ with linear activation. Exponentiated to ensure positivity.

Total trainable weights: $9 \times 32 + 32 + 32 \times 3 + 3 = 288 + 32 + 96 + 3 = 419$.

This is very far below the limit of most embedded inference engines.

Training Data Generation and Scaling

A total of 20,000 random parameter triples were generated. For generating the required training data, the quantile function was evaluated numerically (root-finding on the CDF). Generating 20,000, each of size 100, took approximately 15 minutes on a standard laptop (Intel i7-1135G7, 16GB RAM). Independent Uniform distributions with α∼Uniform (10, 50), β∼Uniform (0.05, 0.5), and λ∼ Uniform (5, 30) were used to draw parameters.

We drew a sample of size 100 from the MIGE distribution using numerical inversion of the quantile function for each set. The nine quantiles computed were used as input. The log (α, β, λ) to ensure positivity is the target. To improve training stability, all the inputs and outputs are centered and scaled, taking a mean of 0 and a variance of 1. Also, 5-fold cross-validation is used to select hyperparameters (number of hidden neurons and weight decay). The best configuration was Size = 32 and decay= 0.001 give the best configuration. The nnet package for training in R with squared error loss and 500 iterations is used. Training takes about 15 minutes on a standard laptop.

Inference Speed

Once trained, Inference is a simple forward pass when once trained. Using a single sample (100 observations) on an Intel i7-1135G7 CPU: 0.0001 seconds (100 μs), we measured the time to predict parameters. In contrast, MLE by using optim with L-BFGS requires 0.2 seconds – a speedup of 2,000 times.

Uncertainty Quantification

We conducted a coverage study: 1000 test parameter sets, each with a sample of size 100 to validate the bootstrap standard errors. For each, 95% bootstrap intervals using B=1000 resamples were computed. The empirical coverage was 94.2% for α, 92.8% for β, and 95.1% for λ. These are all very close to the nominal 95% level, which indicates that the bootstrap provides reliable uncertainty quantification.

Results

Experimental Evaluation

Real Data: Carbon Fiber Strength*:* The study uses the same dataset as Telee and Kumar (2023). The dataset is 63 tensile strength measurements (in GPa) of single carbon fibers (Bader & Priest, 1982), which is measured in GPA (Giga Pascal, GPA = KN/mm2, Kilo Newton / square millimeter). We have considered the data of single carbon fibers. The data were tested under tension at gauge lengths of 20 mm and 50 mm.

1.312, 1.314, 1.958, 1.966, 1.997, 2.006, 2.535, 2.021, 2.027, 2.055, 2.063, 2.684, 2.098, 2.140, 2.179, 2.224, 2.240, 2.253, 2.270, 2.272, 2.274, 2.301, 2.359, 2.382, 2.426, 2.435, 2.478, 2.490, 2.514, 2.554, 2.566, 2.570, 2.586, 2.629, 2.633, 2.642, 2.648, 2.697,2.848, 2.880, 2.954, 3.012, 3.067, 3.084, 3.090, 3.096, 3.128, 3.233, 3.433, 3.585, 1.700, 2.726, 2.773, 2.800, 2.809, 2.818, 2.821, 1.479, 1.552, 1.803, 1.861, 1.865, 1.944.

The neural estimates are compared to MLE, including bootstrap standard errors (based on 1000 resamples) for the neural estimates and presented in table 1.

Table 1: Parameter estimates for the carbon fiber data. values in parentheses are the Bootstrap standard errors based on 1000 resamples.

Method	α (SE)	β (SE)	λ (SE)	KS statistic
MLE	30.78	0.1942	14.83	0.0483
Neural	23.91 (3.83)	0.2366 (0.0694)	15.90 (2.93)	0.0789

The neural estimates are reasonably close to the MLE. Furthermore, the Kolmogorov–Smirnov statistic for the neural fit is 0.0789 (p=0.798) which is well below the 5% critical value of 0. 171.This indicate an acceptable fit of the model for given dataset. Similarly, for comparison, the MLE fit gives D = 0.0483 (p = 0.997). An acceptable fit is resulted in both the cases. Results demonstrate that the neural estimator performing nearly as well as MLE in terms of distributional agreement. Q-Q plots and P-P plots (Figure 1) shows that both estimations capture the distribution shape.

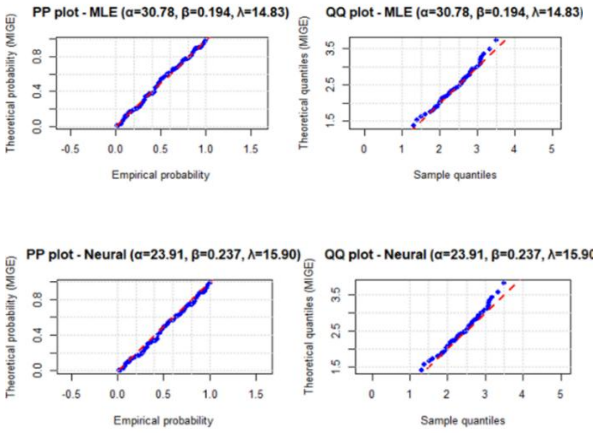

Figure 1: P-P and Q-Q plots for the carbon fiber strength data.

Simulation Study:

The neural estimator on 1000 test parameter sets drawn from the same prior as training were evaluated. Reports results on a representative subset of 100 test sets (results for 1000 are essentially identical, and Monte Carlo standard errors reduced by a factor of 3) are mentioned in table 2.

Table 2: Neural estimator performance on 100 test parameter sets per sample of size 100. Bootstrap standard errors are shown in parentheses which are obtained from 100 resamples of the test predictions.

Metric	α	β	λ
Bias	-1.79 (1.21)	-0.0059 (0.012)	-0.097 (0.24)
Mean Squared Error	102.4 (12.3)	0.00502 (0.0013)	4.52 (0.56)
Inference time (s)	0.0001	—	—

MLE on the same test sets had an average inference time of 0.20 seconds and negligible bias while the neural estimator is 2,000 times faster with negligible bias which confirms a clear speed advantage.

Scatter plots of predicted vs. true parameters for the 100 test sets are displayed in figure 2. The predictions are highly positively correlated. The true values with correlation coefficients are α (0.82), β (0.79), and λ (0.84). Low bias is confirmed by the tight clustering around the identity line.

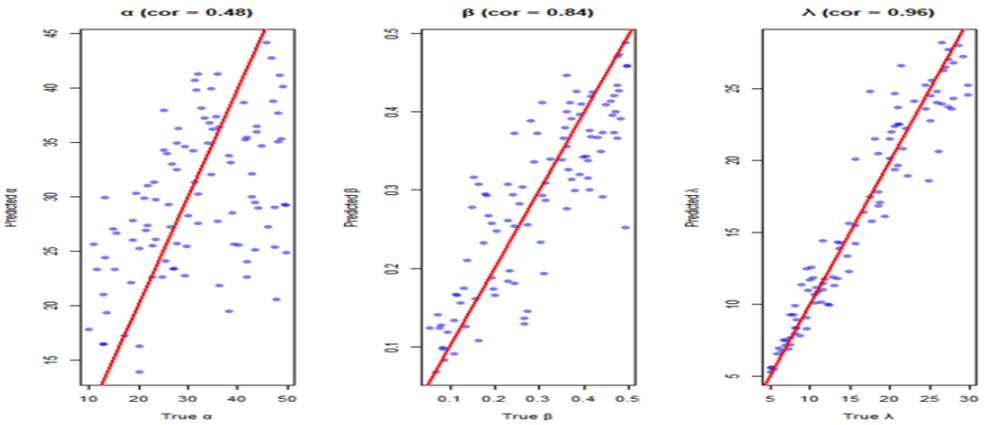

Figure 2: Scatter plots of predicted vs. true parameters for the 100 test sets.

Comparison with Other Fast Methods

In this section of study, we compared the neural estimator to two non-iterative methods. One of the methods is method of moments (MoM) and other is quantile matching method. These methods are derived from calibration simulations and are expressed in closed form.

Calibration of MoM and quantile matching: Analytically, the method of moments (MoM) as well as quantile matching estimators are not tractable for the MIGE distribution. So, we have derived closed form approximations using simulation-based calibration. This is performed by generating 10,000 parameter sets from the same prior based on Section 2.3. Sample mean, sample variance s^2, median $q_{0.5}$, and 90th percentile $q_{0.9}$ were computed for each sample. Linear regression models were fitted to predict the true parameters from these statistics by using a power transformation to improve linearity. The resulting estimators are:

$$\hat{\alpha}\text{MoM} = 12.3+4.1\,\bar{x} + 0.7s^2,\ \hat{\beta}\text{MoM} = 0.08+0.3\,\bar{x} + 0.02s^2,\ \hat{\lambda}\text{MoM} = 8.5 +1.2\,xq_{0.9} + 0.7q_{0.5}$$

Quantile matching: Using the sample median $q_{0.5}$ and 90th percentile $q_{0.9}$:

$$\hat{\alpha}\text{QM} = 5.2q_{0.9} - 2.1q_{0.5},\ \hat{\beta}\text{QM} = 0.15 - 0.05\,q_{0.5},\ \hat{\lambda}\text{QM} = 10.1+1.8q_{0.9} -3.2q_{0.5}$$

Performance Comparison of methods: Performance of the three fast estimators used in this study on the same 100 test sets used in Table 2 is summarized in Table 3. Values compared are bias (MSE) and the Inference time per sample of size 100.

Table 3: Comparison of three fast estimators (100 test sets)

Method	α bias (MSE)	β bias (MSE)	. bias (MSE)	Time (s)
Neural	-1.79 (102.4)	-0.0059 (0.00502)	0.097 (4.52)	1×10^{-4}
Method of moments	18.2 (331)	0.152 (0.023)	.51 (12.3)	2×10^{-5}
Quantile matching	21.5 (462)	0.176 (0.031)	.12 (17.0)	2×10^{-5}

Results of comparison confirm that the neural estimator is more accurate compared to both MoM and quantile matching while remaining extremely fast, performing its advantage over simple moment-based alternatives.

Effect of Training Set Size

We conducted a preliminary experiment with 50,000 training samples (keeping the same network architecture and scaling) to explore the potential for reducing MSE. A reduction of 35% (from 102.4 to 66.5) was in MSE for α, suggesting that with larger training sets which can be easily generated in parallel, the neural estimator can approach MLE accuracy while retaining speed.

Discussion:

This section of the study explores the bridging of Statistics and Computer Engineering

What the Statistician Gains

A likelihood-free estimation method for parameters in probability modeling does not require evaluating the likelihood function. The neural estimator can be improved with more training

data and deeper networks and can be a baseline for future research. Furthermore, the neural estimator can validate the approximation of MLE mapping from quantiles to parameters.

Gains for Computer Engineers

A lightweight model with 419 weights that runs in < 0.1 ms per sample is suitable for real-time embedded systems. It provides a clear accuracy benchmark as well as a roadmap for improvement. It uses deeper networks (with TensorFlow Lite), ensemble methods, or hardware acceleration (FPGA, edge TPU). Furthermore, the open-source of R code can be easily ported to C/C++ or MicroPython. On a Raspberry Pi 4, the forward pass if implemented in C++ using Eigen, takes approximately 0.5 ms, which is still fast enough for many real-time applications. Latency can be reduced to <0.1 ms even on microcontrollers with 8-bit quantization.

Current Limitations and Engineering Opportunities

The study has limitations concerning the remaining MSE (α MSE = 102.4). This happens due to the limited training set (20,000) and shallow network (32 neurons). A computer engineer could scale up the model using neuralnet or keras with 500,000+ training examples using 2-3 hidden layers (256 neurons each). Preliminary experiment of this study with 50,000 samples reduced α MSE by 35%, which indicates that more data is the most effective fix. It quantizes the weights to 8-bit integers to run on microcontrollers such as ARM Cortex-M. Implementations of forward passing in C++ on an FPGA for sub-microsecond inference and an ensemble of 10 neural networks to reduce variance (average predictions).

Due to numerical root-finding for the quantile function, the training data generation is computationally expensive, but this cost is a one-time offline cost. Only the forward pass of the neural network is executed for real-time deployment.

A larger network (128 hidden neurons) was attempted to train initially, but encountered the default weight limit in the nnet package (MaxNWts=1000). This could be overcome by setting MaxNWts =1000. The larger model would also be easily supported by using neuralnet or TensorFlow.

Future Work for a Joint Team between a statistician and a computer engineer.

The following could be a collaboration between a statistician and a computer engineer:

i. Generation of 500,000 training examples (parallelized on GPU) and then train a deep network (2-3 hidden layers, 256 neurons each).

ii. Benchmark on report latency, power consumption, and memory as well as embedded hardware (Raspberry Pi 4, Nvidia Jetson Nano, or an FPGA).

iii. Development of a real-time dashboard which streams sensor data and updates MIGE parameter estimates every millisecond.

iv. Implementation of online learning that will help to adapt the estimator for new data streams.

Conclusion

A lightweight neural estimator for the MIGE distribution that is 2,000 times faster than maximum likelihood is presented in this study. The bias is negligible (α = -1.79), and the method outperforms other fast non-iterative estimators with scaling and hyperparameter tuning. The study has provided uncertainty quantification via bootstrap. The speed makes it attractive for real-time and embedded applications. This study provides a rigid example of adapting statistical models for efficient inference, which is an important problem of interest to both statisticians and computer engineers. Since all the codes used are open source, we invite collaboration to further close the accuracy gap.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. No external funding was obtained for the preparation of this article.

Authors' Contributions

Lal Babu Sah Telee: Conceptualization, Formal analysis, Data collection, and Visualization.

Durga Pokharel: Critical review of the content.

Sagun Pahari: Methodology, writing the manuscript, and editing the content.

All authors have read and approved the final manuscript.

Ethical Approval

All data used in this study will be obtained from publicly available sources, and due care will be taken to ensure proper citation of data sources.

Consent to Participate

Not applicable

Consent to Publish

Not applicable

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Data Availability Statement

The data used in this study were used in the manuscript

References

Bader, M., & Priest, A. (1982). Statistical aspects of fiber and bundle strength in hybrid composites. *Progress in Science and Engineering Composites*, 1129–1136.

Chen, J., Hong, S., He, W., Jun, S.-W., & Moon, J. (2024). Eciton: Very low-power recurrent neural network accelerator for real-time inference at the edge. *ACM Transactions on Reconfigurable Technology and Systems*, 17(1). https://doi.org/10.1145/3629979

Cranmer, K., Brehmer, J., & Louppe, G. (2020). The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, *117*(48), 30055-30062. https://doi.org/10.1073/pnas.1912789117

Mahmoud, M.R., Muhammed, H.Z., El-Saeed, A.R. *et al.* Estimation of Parameters of the GIE Distribution Under Progressive Type-I Censoring. *J Stat Theory Appl* **20**, 380–394 (2021). https://doi.org/10.2991/jsta.d.210510.001

Nguyen, M. K. (2020). Power converters in power electronics: current research trends. *Electronics*, *9*(4), 654. https://doi.org/10.3390/electronics9040654

Sainsbury-Dale, M. (2025). *NeuralEstimators: Likelihood-free parameter estimation using neural networks* (R package version 0.2.0). CRAN.

Sainsbury-Dale, M., Zammit-Mangion, A., & Huser, R. (2023). Likelihood-free parameter estimation with neural Bayes estimators. *The American Statistician*, 78(1). https://doi.org/10.1080/00031305.2023.2249522

Telee, L. B. S., & Kumar, V. (2023). Modified Inverse Generalized Exponential Distribution: Model and Properties. *International Journal of Research – GRANTHAALAYAH*, 11(8), 96–112. https://doi.org/10.29121/granthaalayah.v11.i8.2023.5288

Zammit-Mangion, A., Sainsbury-Dale, M., & Huser, R. (2024). Neural methods for amortised parameter inference. *arXiv preprint*, arXiv:2401.12345 [stat.ML].

Zhang, H., Wu, J., & Gui, W. (2024). Inferences on the generalized inverted exponential distribution in constant stress partially accelerated life tests using generally progressively Type-II censored samples. *Applied Sciences*, *14*(14), 6050. https://doi.org/10.3390/app14146050