Neural Posterior Learning for Bayesian Model Choice

Haowei Zhao

Supervised by Clara Grazian

University of Sydney

1 Abstract

Bayesian statistics provides a powerful way to update beliefs given observed data but in many complex settings, the likelihood functions are intractable and exact computation is impossible. Instead of relying on likelihood functions, Approximate Bayesian Computation (ABC) compares simulated data with the observed data. The traditional method mitigates the curse of dimensionality by using manually selected summary statistics but it may cause information loss.

This report aims at addressing the limitations by using neural network embeddings as summary statistics within the ABC framework. The proposed method is evaluated against full data ABC based on statistical distance and using sufficient statistics in three different experiments. In simple settings, the proposed method achieves classification accuracy, model posterior and parameter estimation variability(across simulations) comparable to those of the benchmark. As complexity increases, the method maintains high classification accuracy and model posterior if partnered with complex architectures but simpler architectures fail to capture subtle data patterns.

The results show a bias-variance trade-off in our proposed method. While the ABC rejection steps can reduce bias in parameter estimation, it sometimes does so at the cost of increased variability across trials. Furthermore, this improvement is not consistent and in some cases, it can lead to a slight increase in variability with no noticeable improvement in accuracy. This suggests that the advantages of our proposed method may be highly dependent on the initial quality of the embeddings extracted from the neural networks.

2 Introduction

2.1 Bayesian Inference

Choosing the right model from several candidate models and estimating model parameters are crucial steps in many fields. Unlike the frequentist approach which treats parameters as fixed but unknown constants, the Bayesian approach sees models and parameters as random variables and assigns probability distributions. It updates prior belief $\pi(\theta)$ after a new set of data y_{obs} is observed and forms the posterior probability $\pi(\theta|y_{obs})$. The whole process is governed by the Bayes' Theorem

$$\pi(\theta|y_{obs}) = \frac{\pi(\theta)p(y_{obs}|\theta)}{p(y_{obs})} \tag{1}$$

where $p(y_{obs}|\theta)$ is the likelihood and $p(y_{obs})$ is the marginal likelihood obtained by integrating $p(y_{obs}|\theta)\pi(\theta)$ over the whole parameter/model space. However, in many complex settings, the likelihood function is intractable and computing the posterior probability explicitly is mathematically infeasible.

2.2 Approximate Bayesian Computation (ABC)

ABC offers a likelihood-free alternative and while there are various ABC methods, they all share the same and simple core idea. Instead of exact computation, ABC simulates data based on parameter priors and compares simulated data to the observed data.

1. Sample a candidate parameter θ' from the prior distribution $P(\theta)$.
2. Generate a simulated dataset X' from the model $M(\theta')$.
3. Accept θ' if the simulated data X' is sufficiently similar to the observed data x_{obs}.
4. The accepted θ' values form empirical distribution of the model parameter θ

The same principle applies to model choice. Although this is a powerful framework to approximate the posterior probability, simulations are far less likely to fall close to the observed data in higher dimension. This is known as the 'Curse of Dimensionality' and may lead to a failure of the above-mentioned rejection algorithm.

To mitigate this, ABC traditionally relies on the use of summary statistics rather than the whole data. But it may introduce information loss if the summary statistics are not 'sufficient' enough to fully capture the relevant information about the parameter/model[Robert et al., 2011].

Recent advances in ABC promoted an alternative way which uses full simulated and observed data and statistical distances, such as MMD[Park et al., 2015] and Wasserstein[Bernton et al., 2019], to perform comparisons between empirical distributions. A systematic comparison revealed that using this full data ABC based on statistical distances yields posteriors that closely match the results from ABC using sufficient statistics or provides high support for the true model[Grazian, 2025]. While such an approach will serve as useful comparisons in this report, the main focus is to utilise the power of deep learning to learn summary statistics. By training a neural network for both classification and regression tasks, it learns low dimensional but highly informative statistics to be used within the ABC framework.

2.3 Neural Network

The idea behind deep learning started with neural networks which were inspired by human brains[Hopfield, 1982]. A neural network is a mathematical model that consists of an input layer, one or more hidden layers and an output layer. The layers are designed to approximate functions and identify important features of the data. Model parameters are optimised from training data by minimizing a loss function. Due to its strong ability to recognise complex patterns of data, neural networks are widely used for classification and regression across various fields.

In population genetics, deep learning has been used to infer natural selection and demographic history[Sheehan and Song, 2016]. Similarly in organic chemistry, neural networks have been applied to kinetic data to identify the most probable underlying reaction mechanism without deriving rate laws and thus reduces human error[Burés and Larrosa, 2023]. This technology is also increasingly welcomed in medical diagnostics where one dimensional convolutional neural network has demonstrated its ability to classify cardiac arrhythmias rapidly and accurately based on ECG data[Mishra et al., 2024][Sodagi et al., 2024]. In this report, we explore the idea of using neural network embeddings within the ABC framework and provide a comparative analysis against ABC based on sufficient statistics and a statistical distance metric. The code used in this report is available on GitHub at: `https://github.com/Wildseventeen/NN-ABC-MODEL-CHOICE`.”

3 Statement of Authorship

The methodology of using neural network embeddings within the ABC framework and the code to implement ABC algorithm employed in this report were primarily developed by Prof Clara

Grazian. Of the three experiments presented, the first two were conducted using simulation and analysis code from her previous work. For the third experiment, the idea came from [Drovandi and Frazier, 2022]. Simulation code was independently developed while the subsequent analysis was performed using code developed by Prof Grazian. The data interpretation and the comparative performance analysis among neural network, using neural network embeddings within ABC, ABC with sufficient statistics and full data ABC were conducted collaboratively by Prof Grazian and me.

4 Methodology

4.1 ABC Algorithm

We modified Algorithm 4 (Summary-Based ABC for Model Choice) in [Grazian, 2025] to incorporate neural network embeddings. This will be referred to as NN-ABC in later sections in this report.

Algorithm 1 ABC-MC based on Neural Network Embeddings(NN-ABC)

```
1: **Input:** Given observed dataset $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$, a set of candidate models $\{M_1, \ldots, M_K\}$, a trained
   neural network $f(\cdot)$, a tolerance level $\varepsilon$, and Euclidean distance $\|\cdot\|_2$.
2: **Summary statistics of observed:** Pass observed data $\mathbf{y}$ through $f(\cdot)$ and extract the activations from
   the penultimate layer $\mathbf{e}_{obs} = f_{layer_{-1}}(\mathbf{y})$.
3: for $i = 1, \ldots, N$ do
4:     repeat
5:         Generate $M_{k^*}$ from prior $\pi(M_k)$.
6:         Generate parameters $\boldsymbol{\theta}_{k^*}$ from prior $\pi_{k^*}(\boldsymbol{\theta}_{k^*})$.
7:         Simulate dataset $\mathbf{z}$ from the chosen model $M_{k^*}$ with parameters $\boldsymbol{\theta}_{k^*}$.
8:         Pass $\mathbf{z}$ through $f(\cdot)$ and extract the activations from the penultimate layer $\mathbf{e}_{sim} = f_{layer_{-1}}(\mathbf{z})$.
9:     until $\|\mathbf{e}_{obs} - \mathbf{e}_{sim}\|_2 \leq \varepsilon$
10:    Set $M^{(i)} = M_{k^*}$ and $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}_{k^*}$.
11: end for
12: **Output:** A set of values $(M^{(1)}, \ldots, M^{(N)})$ and $(\theta^{(1)}, \ldots, \theta^{(N)})$ from $\pi_\varepsilon(M_k \mid f_{layer_{-1}}(\mathbf{y}))$ and $\pi_{\varepsilon,k}(\theta_k \mid$
    $f_{layer_{-1}}(\mathbf{y}), M_k)$, for $k = 1, \ldots, K$, respectively.
```

To approximate the model and parameter posteriors, a total of 10^6 reference datasets are simulated for each experiment. Rather than picking a fixed value for ε, we follow the approach suggested in [Biau et al., 2013] that only the closest $q\%$ of simulations will be retained and we choose $q\% = 0.1\%$.

4.2 Neural Network Architectures

The performance of the NN-ABC method is evaluated using five neural network architectures from simple feed-forward to more complex convolutional neural networks. Their specific structures and hyper-parameters were adopted from publications where they had been applied to some real-world problems. The embedding dimension that is defined by the number of activations in the penultimate layer varies across the five architectures. For example, Neural Network 1 uses a 64-dimensional embedding, while Neural Network 2 uses 10 dimensions. While these dimensions differ between architectures, the structure of each individual network remains fixed across all three experiments. See Appendix A for more details about architectures.

4.3 Neural Network Training

The training strategy mimics that in [Grazian, 2025]. A neural network is trained on 10^6 datasets separately generated from model and parameter priors to reduce overfitting. The training set is split into 80% for training and 20% for validation to monitor performance during the optimization process.

The network performs multi-task learning to optimise two objectives simultaneously. Model selection is via classification by using categorical cross-entropy loss function and parameter estimation is via regression by minimizing the MSE loss. For Experiment 3, we implemented a customised masked MSE loss function which ensures that backpropagation only occurs for parameters relevant to a specific model.

To balance these two objectives, the total loss function is defined as a weighted sum of the two objectives. The classification weight is set to 1.0 while the regression weight is set to 0.5, giving more importance to model selection. In experiment 2 and 3 where data are right skewed, we apply *arcsinh* to normalise the data. Neural networks are trained separately for each experiment.

5 Experiments and Results

Each experiment is conducted over 100 independent trials for each competing model. Non-informative uniform priors are assigned as model priors across all experiments while parameter priors are specified individually. For ABC, a reference set of 10^6 simulations is generated and evenly distributed across candidate models.

NN-ABC performance is compared with Neural Network(NN), ABC using sufficient statistics(ABC-Stat), if exist and full data ABC based on Wasserstein distance metric(ABC-Wass). This statistical distance has been shown to work well in the first two experiments by Grazian [2025] and on the $M/G/1$ model in the third experiment by [Drovandi and Frazier, 2022] when applied to log transformed data. When running ABC with sufficient statistics, no information is lost and we employ Euclidean distance to run the ABC algorithm. This will serve as a benchmark.

5.1 Experiment 1: Normal Mean Hypothesis Test

5.1.1 Experimental Setup

We follow the same experimental setup from [Grazian, 2025]. In this case study, data are sampled from normal distributions $N(\theta, 1)$ and we consider two competing hypotheses: null model H_0 where the mean θ is fixed at 0 and alternative model H_1 where θ follows a prior distribution of $N(0, 100)$.

In this simple setting, the sample mean is a sufficient statistic for θ. To evaluate the performance of the NN-ABC method, 100 test datasets are generated from $N(\theta_0, 1)$ for each $\theta_0 \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. When $\theta_0 = 0$, the data-generating model is H_0, while for non-zero θ_0, the test datasets are drawn from the alternative H_1. As shown in the density plots Figure 1, the classification task is most challenging when θ_0 is close to 0 due to the high degree of overlap between the null and alternative. As θ_0 departs from zero, the plots become increasingly distinct.

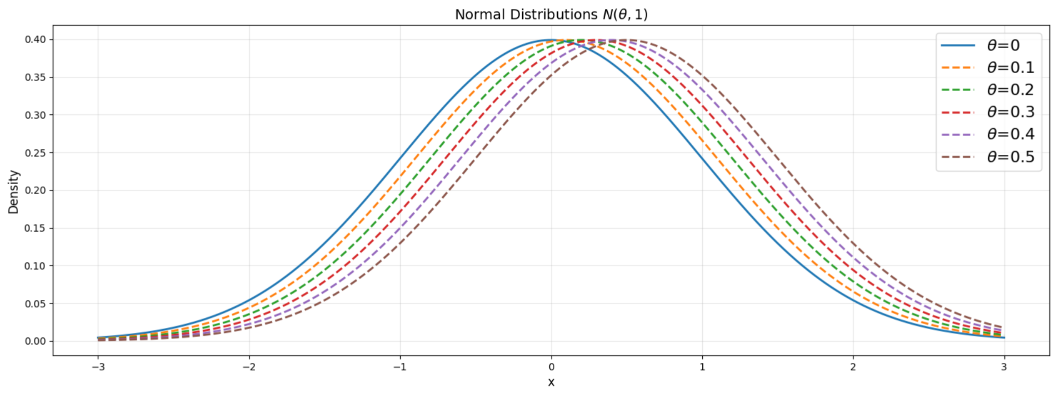


Figure 1: Probability density functions for the normal distribution $N(\theta, 1)$

5.1.2 Results

We first assess the performance on model choice. Methods are applied to each of the test sets to estimate the probability of choosing null hypothesis H_0 and we report the average across the 100 test sets in Table 1.

To evaluate classification accuracy, we select the model (H_0 or H_1) that yields the higher posterior probability for each individual test set. The results of these 100 decisions per θ_0 are then aggregated into the confusion matrices presented in Appendix B. This approach allows us to see the frequency of correct classifications versus Type I and Type II errors across the different methods.

		Neural Network					**NN-ABC**					**ABC**	
θ_0		NN1	NN2	NN3	NN4	NN5	NN1	NN2	NN3	NN4	NN5	Stat	Wass
0.0	$\Pr(\theta = 0)$	0.970	0.970	0.888	0.973	0.964	0.969	0.971	0.914	0.974	0.971	0.972	0.972
0.1	$\Pr(\theta = 0)$	0.937	0.928	0.886	0.936	0.915	0.937	0.930	0.912	0.933	0.935	0.927	0.935
0.2	$\Pr(\theta = 0)$	0.779	0.779	0.885	0.813	0.756	0.790	0.775	0.909	0.808	0.811	0.761	0.792
0.3	$\Pr(\theta = 0)$	0.541	0.479	0.880	0.523	0.462	0.561	0.477	0.886	0.511	0.555	0.455	0.507
0.4	$\Pr(\theta = 0)$	0.256	0.229	0.878	0.262	0.199	0.276	0.229	0.880	0.251	0.275	0.199	0.248
0.5	$\Pr(\theta = 0)$	0.064	0.056	0.871	0.059	0.054	0.073	0.054	0.836	0.050	0.095	0.040	0.063

Table 1: Average Posterior Probability of H_0

As anticipated, when the true mean θ_0 moves further away from 0, all methods except for NN3 and its NN-ABC counterpart gradually shift their support from H_0 to H_1. When H_0 is the true data-generating model, all methods identify H_0 near-perfectly. However, NN3 does not seem to respond to the change in the data as θ_0 increases. Overall, most methods besides architecture 3 perform well and achieve results comparable to the ABC-Stat benchmark.

We further assess parameter estimation performance using the boxplots of the posterior mean of θ. See Appendix B for boxplots. NN3 and its NN-ABC version are excluded as they clearly fail to recognise H_1. When $\theta_0 = 0, 0.4$, and 0.5, all methods are generally at their best for identifying the true data-generating model. However, the boxplots show that using the neural network alone leads to larger variability and more biased results. In contrast, the NN-ABC and ABC-Wass methods produce boxplots that are more tightly centred around the true value, matching the performance of the benchmark ABC-Stat.

5.2 Experiment 2: Exponential Family

5.2.1 Experimental Setup

This section presents a more complex example concerning three models from the exponential family. We follow the same setup as in [Marin et al., 2016]. Under Model 1, the data are modeled as i.i.d. samples from Exp(θ), where the rate parameter θ has an Exp(1) prior. Under Model 2, the data follow a Log-Normal distribution Lognormal($\theta, 1$), where $\theta \sim N(0, 1)$ is the underlying normal mean and the dispersion parameter is set to 1. Model 3 assumes a Gamma($2, \theta$) distribution, where the rate parameter follows an Exp(1) prior. In this setting, the vector $\left(\sum_{i=1}^{n} y_i, \sum_{i=1}^{n} \log(y_i), \sum_{i=1}^{n} \log^2(y_i)\right)$ is shown to be sufficient for the problem of model choice in the sense of the paper by[Marin et al., 2016]. Test sets are drawn from Exp(1), Lognormal($0.193, 1$), and Gamma($2, 1$). These θ values were chosen so that all three distributions share the same mean and have a highly overlapping model space as shown in Figure 2[Grazian, 2025].

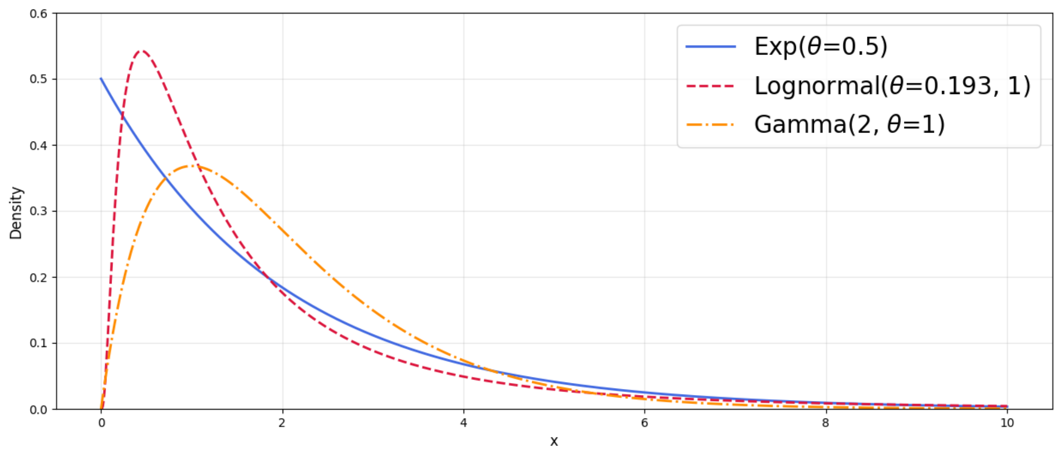


Figure 2: Probability density functions for Exp(0.5), Lognormal$(0.193, 1)$ and Gamma$(2, 1)$

5.2.2 Results

Results are summarised in Table 2 and Appendix C.

		Neural Network					NN-ABC					ABC		
		NN1	NN2	NN3	NN4	NN5	NN1	NN2	NN3	NN4	NN5	Stat	Wass	Wass(log)
Exp(0.5)	$\Pr(M = M_1)$	0.434	0.380	0.457	0.951	0.944	0.456	0.380	0.480	0.942	0.938	0.912	0.808	0.912
Lognormal$(0.193, 1)$	$\Pr(M = M_2)$	0.560	0.404	0.593	0.953	0.972	0.540	0.416	0.579	0.962	0.958	0.942	0.825	0.943
Gamma$(2, 1)$	$\Pr(M = M_3)$	0.951	0.639	0.588	0.984	0.987	0.942	0.629	0.624	0.977	0.983	0.956	0.985	0.983

Table 2: Average Posterior Probability of Correctly Selecting the True Model

Given the similarity of all three models, identifying the true model is more difficult than the previous experiment. The benchmark ABC-Stat and ABC-Wass(log) achieve over 90% in both classification accuracy and posterior in all cases while ABC-Wass is slightly less satisfactory because the data are right skewed. Among the neural networks, NN4 and NN5 are the top performers and they closely match the benchmark with very high classification accuracy and posterior regardless of being used alone or combined with ABC. In contrast, NN1, NN2 and NN3 really struggle with test data coming from exponential and log-normal. The posteriors are just around 0.5. Furthermore, incorporating their embeddings in ABC does not noticeably improve their performance. This suggests that simple neural networks may not be able to handle high overlap of competing models while more complex convolutional neural networks are as effective as ABC based on sufficient statistics.

For the parameter estimation results, we exclude NN1, NN2 and NN3. Looking at the boxplots in Appendix C, both the ABC-Wass and ABC-Wass(log) methods perform well with boxplots that look almost identical to the benchmark ABC-Stat across all three cases. In general, the neural network methods show more bias in their estimates. When the true model is Exp(0.5), the NN-ABC approach helps by providing more accurate estimates than the neural network on its own. This is shown in the boxplots where the NN-ABC results are more tightly grouped with medians being closer to the true value. However, for M_2 and M_3, we notice the bias-variance tradeoff. In these instances, the ABC steps fix the bias coming from the neural network but this leads to more spread-out boxplots.

5.3 Experiment 3: Queuing Systems

A queuing system refers to any scenario where customers arrive, wait in line, receive service, and then leave. Two major components are the arrival process of customers and the service mechanism. We assume first come first served for queue rule.

5.3.1 Experimental Setup

We adapted a setup very similar to that in [Drovandi and Frazier, 2022] but in addition to $M_1 : M/G/1$, we have two more models $M_2 : M/M/1$ and $M_3 : M/D/1$. While all three models assume that inter-arrival times follow exponential distribution Exp(θ_1), they differ in the service time distributions. M_1 uses a uniform distribution U(θ_2, θ_3), M_2 follows Exp(θ_4) and M_3 uses constant service time θ_5. The model parameters are assigned uniform priors as detailed in Table 3.

	M_1: M/G/1	**M_2: M/M/1**	**M_3: M/D/1**
Inter-arrival	Exp(θ_1)	Exp(θ_1)	Exp(θ_1)
Service Time	$U(\theta_2, \theta_3)$	Exp(θ_4)	Constant θ_5
θ_1 **Prior**	$U(0, 0.4)$	$U(0, 0.4)$	$U(0, 0.4)$
θ_2 **Prior**	$U(0, 4)$	–	–
θ_3 **Prior**	$U(0, 8)$	–	–
θ_4 **Prior**	–	$U(0, 0.6)$	–
θ_5 **Prior**	–	–	$U(0, 3)$

Table 3: θ Priors

For this experiment, we use the inter-departure times as the observed data to perform model selection and parameter inference. The ABC reference data and neural network training data are simulated subject to the condition that the traffic intensity $\rho = \frac{\text{arrival rate}}{\text{service rate}} < 1$ to ensure the queue is able to reach steady state. This constraint is necessary because if the arrival rate exceeds the service rate, the queue length would grow indefinitely and the inter-departure times would eventually equal the service times and contain very little information about arrival rate θ_1.

Because there are no known sufficient statistics for these queuing model parameters, we rely on the findings from [Drovandi and Frazier, 2022] which concludes that the Wasserstein distance with log-transformed data is the most effective metric on $M/G/1$ model. To evaluate performance, we simulate 100 test datasets from each of the three models using the following θ values:

- M_1: Inter-arrival $\sim$ Exp(0.2) and Service Time $\sim$ U(1, 5)
- M_1: Inter-arrival $\sim$ Exp(0.1) and Service Time $\sim$ Exp(0.35)
- M_1: Inter-arrival $\sim$ Exp(0.3) and Service Time = 2

5.3.2 Results

The performance is summarised in Table 4 with detailed classification accuracy shown in the confusion matrices in Appendix D. Same as the previous experiment, the results show a clear divide between the simpler neural network architectures and the more complex ones when dealing with this more challenging problem.

		Neural Network					**NN-ABC**					**ABC**
		NN1	NN2	NN3	NN4	NN5	NN1	NN2	NN3	NN4	NN5	Wass(log)
$M/G/1$	$\Pr(M = M_1)$	0.512	0.461	0.552	0.996	0.998	0.525	0.455	0.551	0.992	1.000	1.000
$M/M/1$	$\Pr(M = M_2)$	0.563	0.467	0.754	0.822	0.857	0.609	0.464	0.690	0.855	0.842	0.832
$M/D/1$	$\Pr(M = M_3)$	0.661	0.597	0.469	0.760	0.998	0.584	0.581	0.599	0.955	0.993	0.952

Table 4: Average Posterior Probability of Correctly Selecting the True Model

ABC-Wass(log), NN4 and NN5 and their corresponding NN-ABC method perform very well at identifying M_1 and M_3 with posterior probabilities and classification accuracy over 0.9. NN4 is slightly less certain at 0.760 for M_3.

M_2 appears to be the most difficult model for all methods. The average posteriors for this model are around 0.85 even for the best performing methods. While the NN-ABC approach seems to help the simple architectures (NN1, 2 and 3) in some cases, this improvement is not guaranteed and can sometimes also lead to a slight decrease in performance.

Turning to the estimation variability, we exclude architectures NN1, NN2, and NN3 along with their NN-ABC version because of their underperformance in classification. Based on the boxplots in Appendix D, the ABC-Wass(log) method is no doubt the most effective method as it consistently produces boxplots with medians that sit almost exactly on the true values and also with the smallest variability. The performance of the NN-ABC approach varies in this experiment. In some cases, we observe the best of both. For example, when estimating θ_4 for the $M/M/1$ model, the ABC steps not only correct the estimates from the neural network but also reduce the overall variability. However, most of the time, more accurate estimates come with a larger spread in the boxplots. There are also cases where the ABC steps do not seem to help much at all. When estimating θ_2 in the M/G/1 model, the NN-ABC architecture 5 just shifts the NN5 estimates from an underestimate to an overestimate without actually improving the accuracy or reducing the spread.

6 Discussion and Conclusion

This report has explored the idea of using neural network embeddings within the summary-based ABC algorithm for the purpose of mitigating information loss associated with manual statistic selection. By testing different architectures across three experiments, we observed strengths and limitations of this NN-ABC approach.

The results show that the choice of architecture is one of the most critical factors. While almost all methods performed well in the simplest experiment, the deeper convolutional networks are much more effective when the problem becomes more complex. NN3 failed in almost every case probably because of its use of one single convolutional filter. It may have limited its ability to extract meaningful features from the raw data. Meanwhile, the simple FFNN performed well in the first experiment but failed in the second and third. This is likely because these experiments involve complex and overlapping model space, which requires an architecture capable of extracting spatial or temporal patterns. Simple FFNN is not designed for tasks with this level of difficulty.

Another observation across all experiments is that the performance of the NN-ABC was tied to the performance of the neural network. When the standalone NN performed well, the NN-ABC version also excelled. And when the NN failed, the ABC steps would not help much. This is possibly because the ABC rejection algorithm relies heavily on the quality of the summary statistics (embeddings) obtained from the neural network. This reaffirms that which architecture works the best is problem-specific and choosing an appropriate network structure is the primary factor of success for this NN-ABC method.

Regarding parameter estimation, the advantages of employing NN-ABC are likely dependent on the complexity of the problem. In the first and simplest experiment, NN-ABC produced boxplots with smaller variability than standalone neural networks. The boxplots appeared almost identical to those from the benchmark. However, in more complex scenarios, a bias-variance trade-off became apparent. The subsequent ABC steps often corrected biased estimates from the neural network at the cost of larger variability. There are also cases when the ABC steps neither improved the accuracy nor reduced variability. This indicates that the degree of benefits obtained from the additional ABC steps in the NN-ABC method is again related to the quality of embeddings. Although architectures 4 and 5 performed well in model selection in the third experiment, inter-departure times are a time series themselves. Then recurrent architectures such as LSTM may be better suited to capture sequential dependencies. Furthermore, the training process placed much more weight on classification. This may have led to embeddings becoming more informative for model choice rather than parameter inference and contribute to the inconsistent benefits we gained from NN-ABC.

Apart from exploring more advanced architectures, another direction for assessing the NN-ABC method is the efficiency of the ABC rejection steps. While full data ABC generally requires a strict threshold such as 0.1% used in this report, the NN-ABC may allow for a less strict threshold without decreasing the accuracy for model selection. This may show whether NN-ABC offers a computational advantage compared to the full data ABC approach.

A Neural Network Architectures

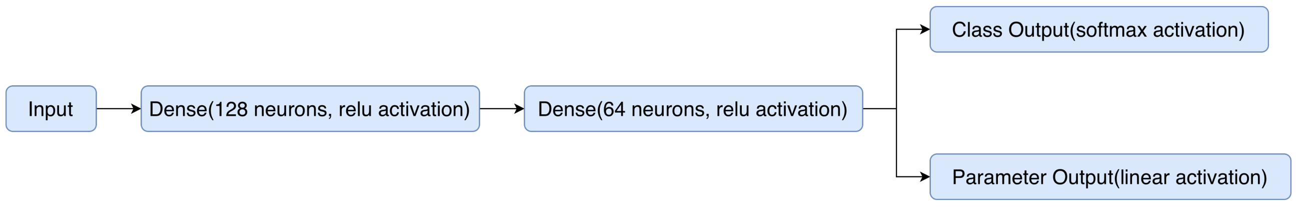

Input
Dense(128 neurons, relu activation)
Dense(64 neurons, relu activation)
Class Output(softmax activation)
Parameter Output(linear activation)

Figure A.1: Architecture of Neural Network 1

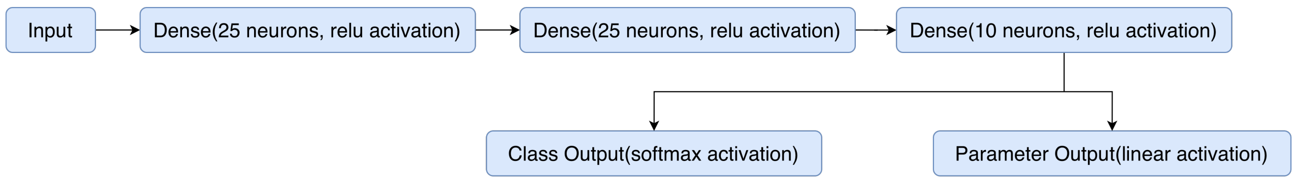

Input
Dense(25 neurons, relu activation)
Dense(25 neurons, relu activation)
Dense(10 neurons, relu activation)
Class Output(softmax activation)
Parameter Output(linear activation)

Figure A.2: Architecture of Neural Network 2

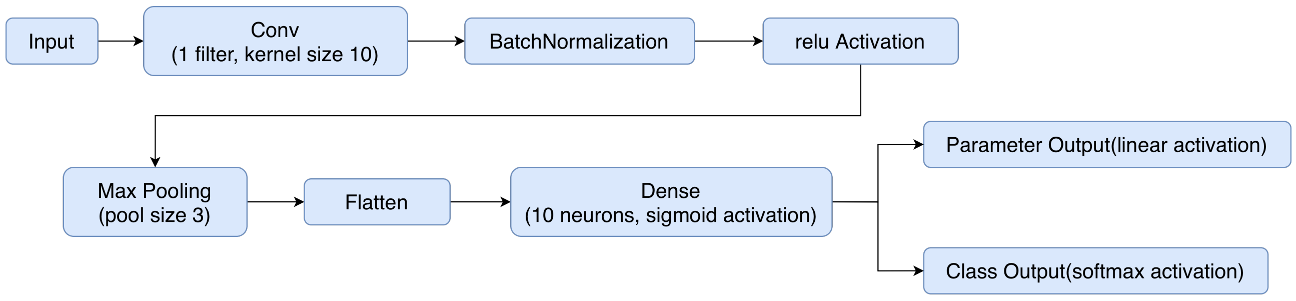

Input
Conv (1 filter, kernel size 10)
BatchNormalization
relu Activation
Max Pooling (pool size 3)
Flatten
Dense (10 neurons, sigmoid activation)
Parameter Output(linear activation)
Class Output(softmax activation)

Figure A.3: Architecture of Neural Network 3

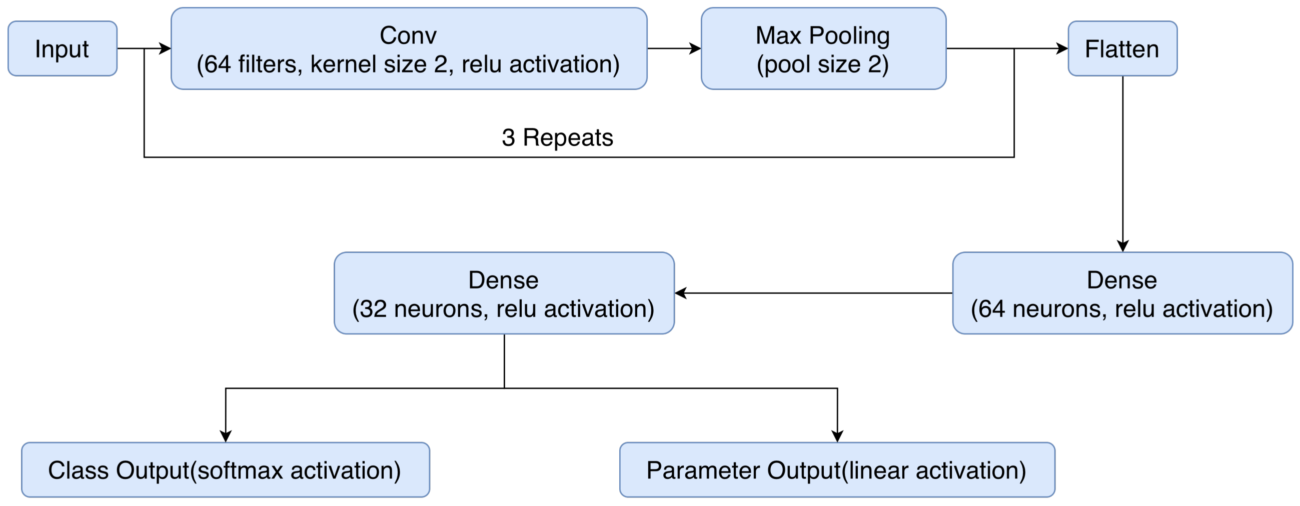

Input
Conv
(64 filters, kernel size 2, relu activation)
Max Pooling
(pool size 2)
Flatten
3 Repeats
Dense
(32 neurons, relu activation)
Dense
(64 neurons, relu activation)
Class Output(softmax activation)
Parameter Output(linear activation)

Figure A.4: Architecture of Neural Network 4

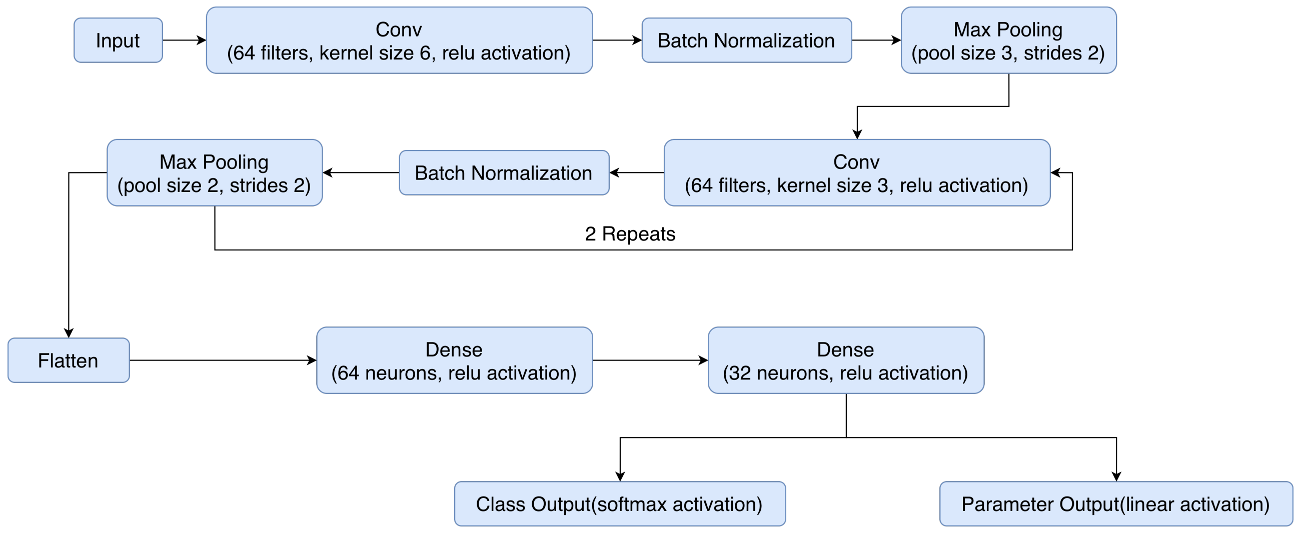
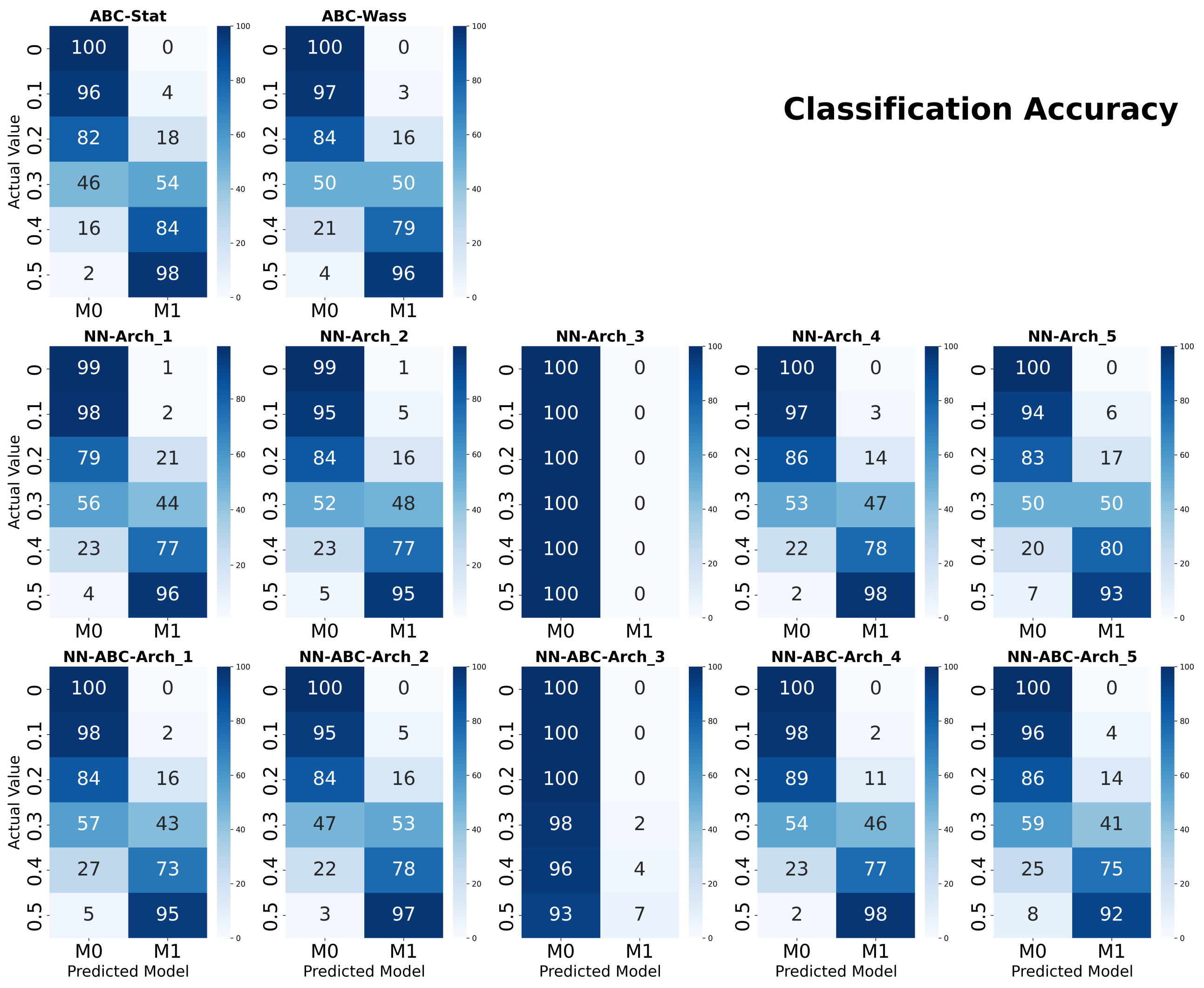

Input
Conv
(64 filters, kernel size 6, relu activation)
Batch Normalization
Max Pooling
(pool size 3, strides 2)
Max Pooling
(pool size 2, strides 2)
Batch Normalization
Conv
(64 filters, kernel size 3, relu activation)
2 Repeats
Flatten
Dense
(64 neurons, relu activation)
Dense
(32 neurons, relu activation)
Class Output(softmax activation)
Parameter Output(linear activation)

Figure A.5: Architecture of Neural Network 5

B Normal Mean Hypothesis Test



Actual Value	ABC-Stat M0	ABC-Stat M1	ABC-Wass M0	ABC-Wass M1
0	100	0	100	0
0.1	96	4	97	3
0.2	82	18	84	16
0.3	46	54	50	50
0.4	16	84	21	79
0.5	2	98	4	96

Actual Value	NN-Arch_1 M0	NN-Arch_1 M1	NN-Arch_2 M0	NN-Arch_2 M1	NN-Arch_3 M0	NN-Arch_3 M1	NN-Arch_4 M0	NN-Arch_4 M1	NN-Arch_5 M0	NN-Arch_5 M1
0	99	1	99	1	100	0	100	0	100	0
0.1	98	2	95	5	100	0	97	3	94	6
0.2	79	21	84	16	100	0	86	14	83	17
0.3	56	44	52	48	100	0	53	47	50	50
0.4	23	77	23	77	100	0	22	78	20	80
0.5	4	96	5	95	100	0	2	98	7	93

Actual Value	NN-ABC-Arch_1 M0	NN-ABC-Arch_1 M1	NN-ABC-Arch_2 M0	NN-ABC-Arch_2 M1	NN-ABC-Arch_3 M0	NN-ABC-Arch_3 M1	NN-ABC-Arch_4 M0	NN-ABC-Arch_4 M1	NN-ABC-Arch_5 M0	NN-ABC-Arch_5 M1
0	100	0	100	0	100	0	100	0	100	0
0.1	98	2	95	5	100	0	98	2	96	4
0.2	84	16	84	16	100	0	89	11	86	14
0.3	57	43	47	53	98	2	54	46	59	41
0.4	27	73	22	78	96	4	23	77	25	75
0.5	5	95	3	97	93	7	2	98	8	92

Figure B.1: Classification Accuracy: Normal Mean

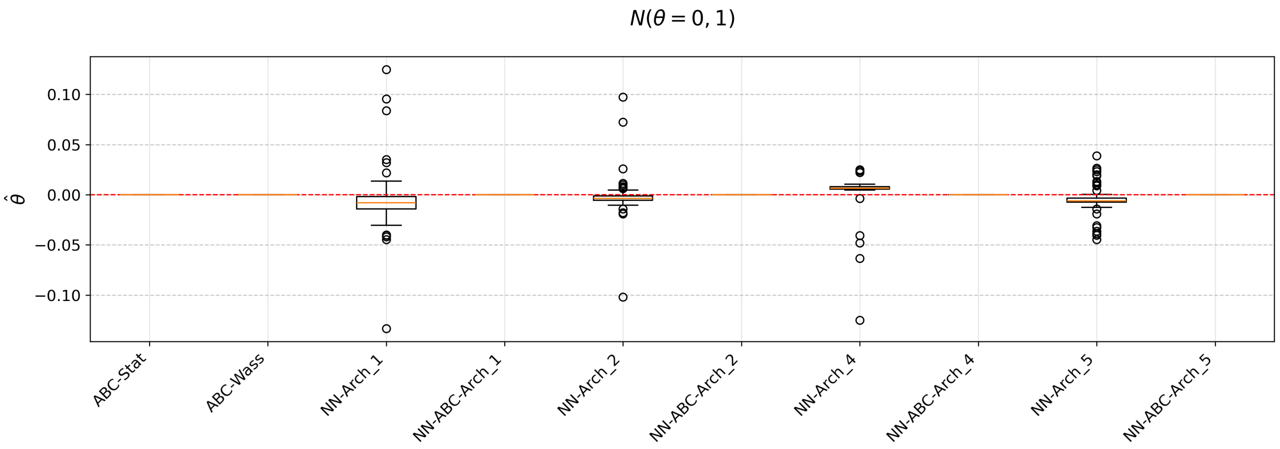


Figure B.2: Estimated posterior means for θ across 100 independent trials (true value $\theta = 0$)

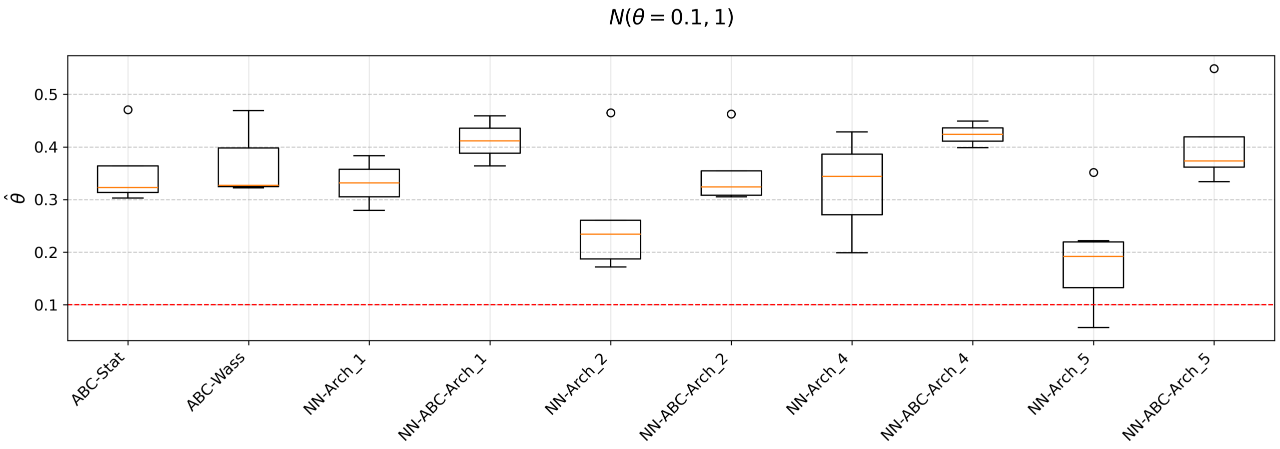


Figure B.3: Estimated posterior means for θ across 100 independent trials (true value $\theta = 0.1$)

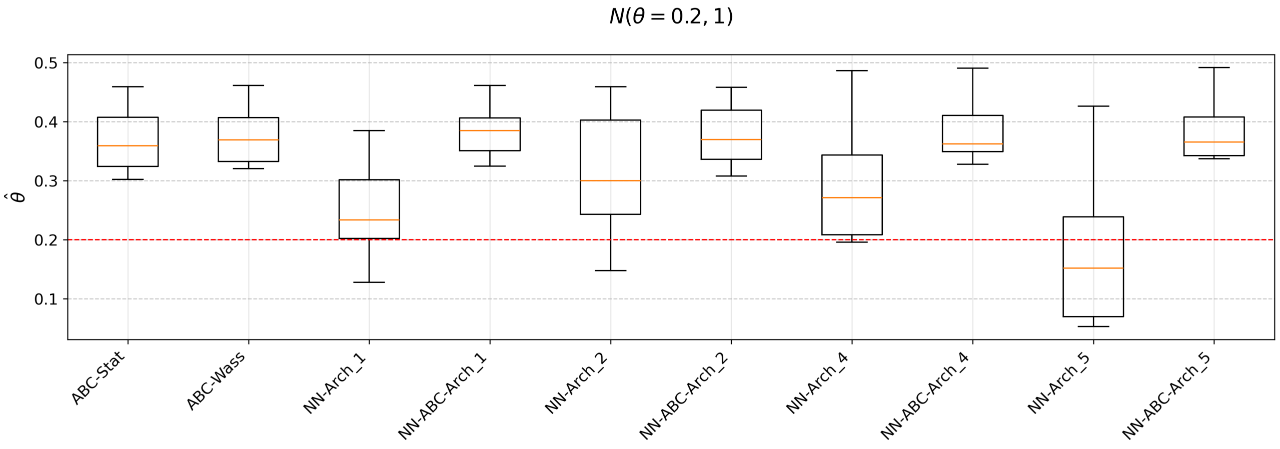


Figure B.4: Estimated posterior means for θ across 100 independent trials (true value $\theta = 0.2$)

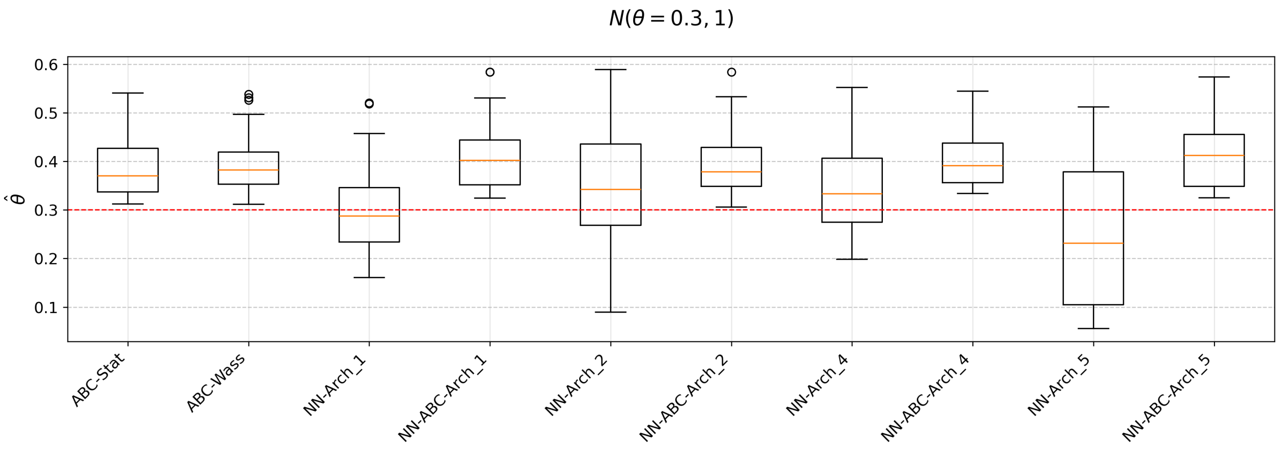


Figure B.5: Estimated posterior means for θ across 100 independent trials (true value $\theta = 0.3$)

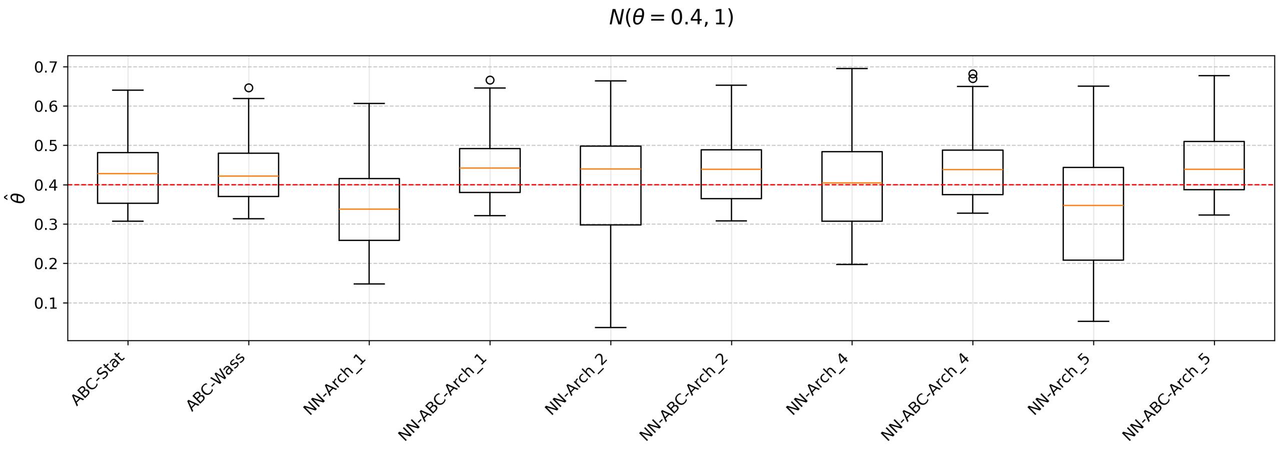


Figure B.6: Estimated posterior means for θ across 100 independent trials (true value $\theta = 0.4$)

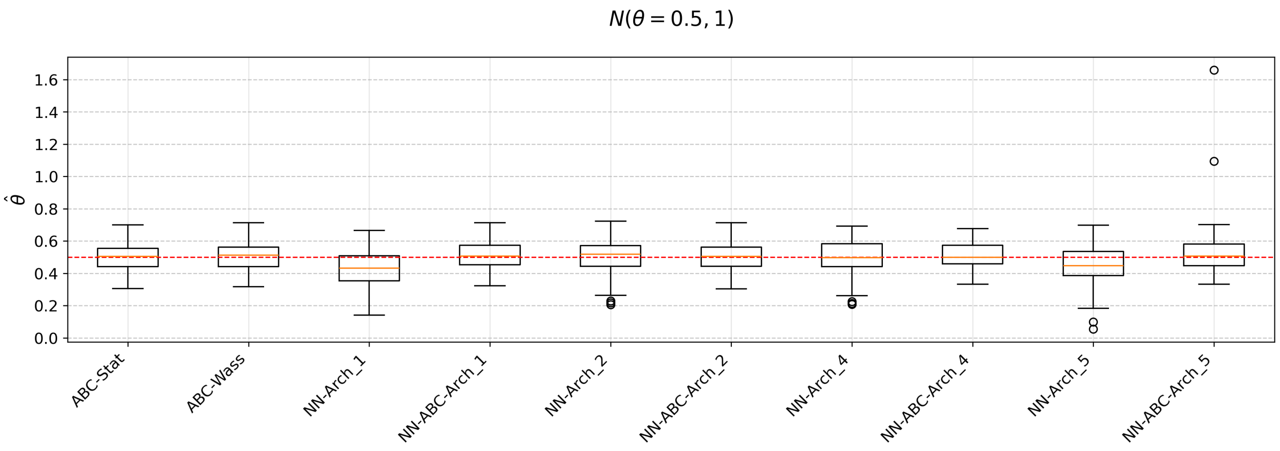


Figure B.7: Estimated posterior means for θ across 100 independent trials (true value $\theta = 0.5$)

C Exponential Family

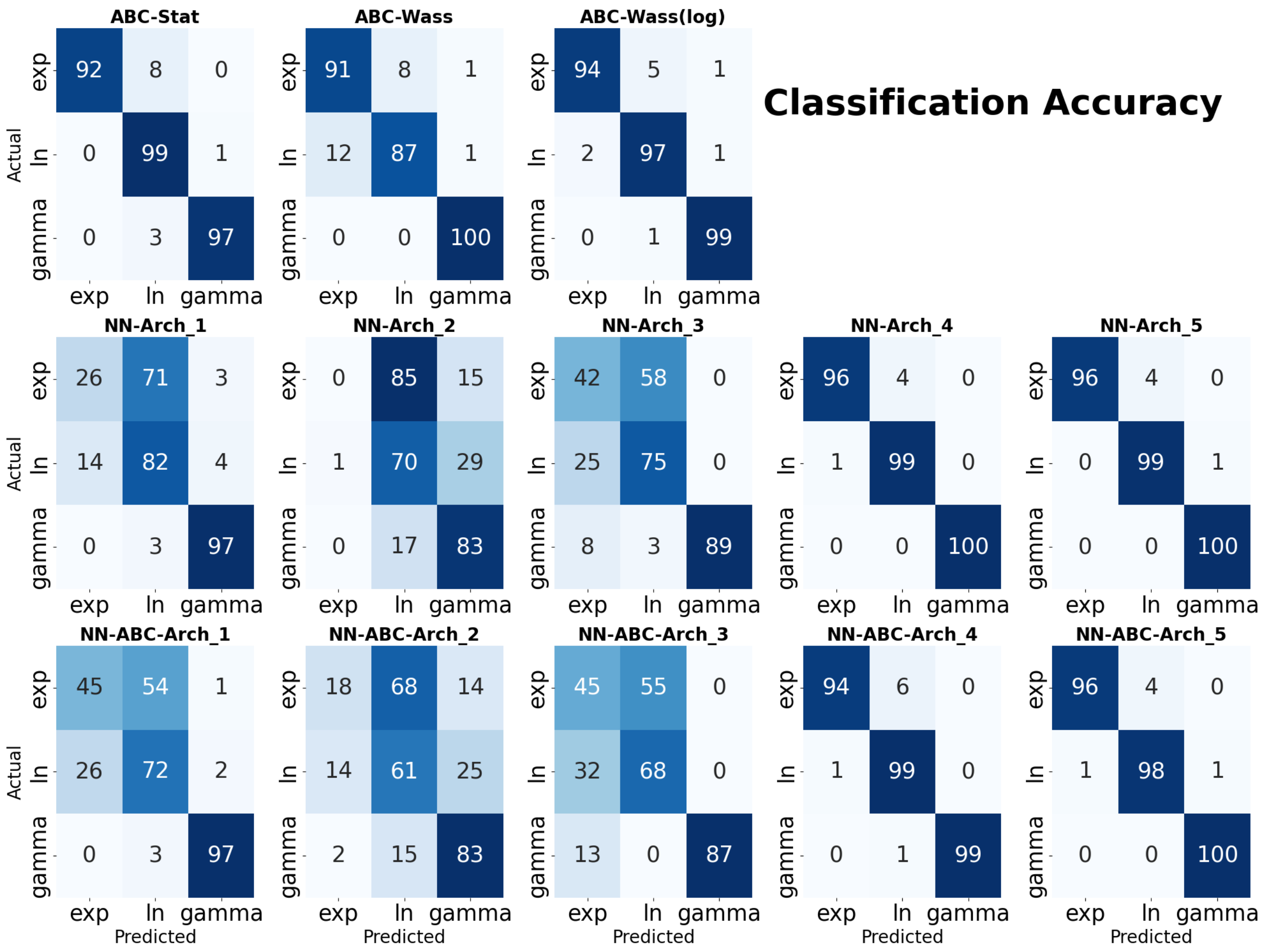


Figure C.1: Classification Accuracy: Exponential Family

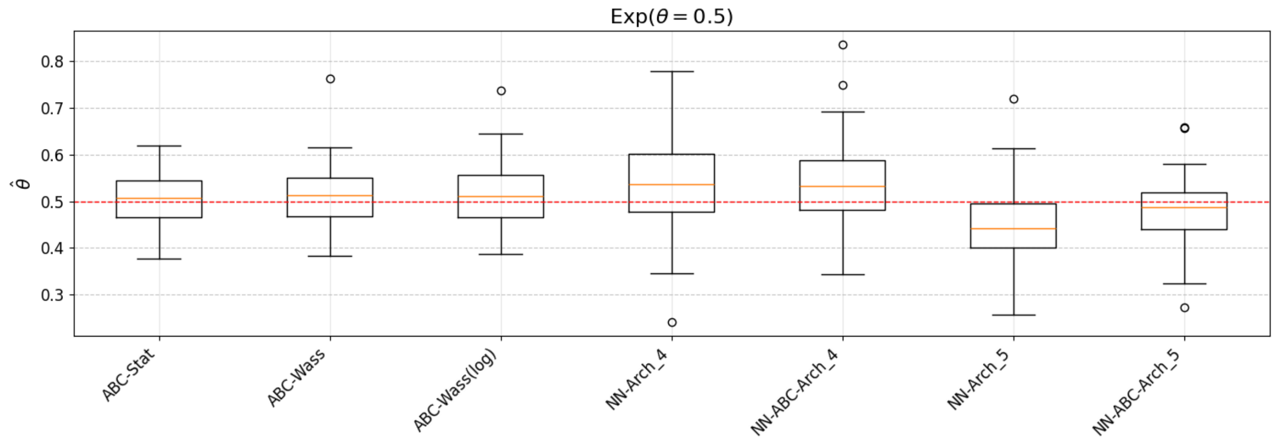


Figure C.2: Estimated posterior means for θ across 100 independent trials (true value $\theta = 0.5$)

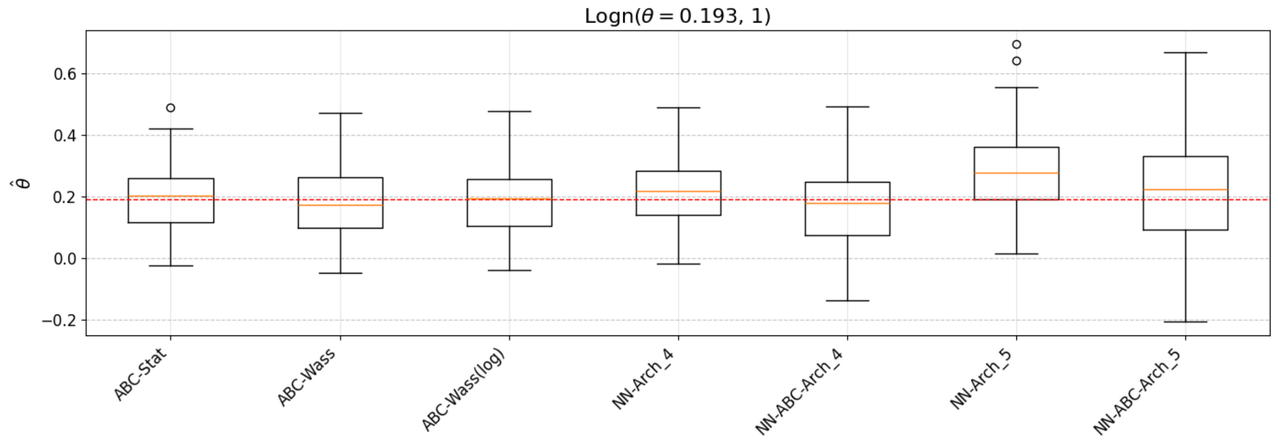


Figure C.3: Estimated posterior means for θ across 100 independent trials (true value $\theta = 0.193$)

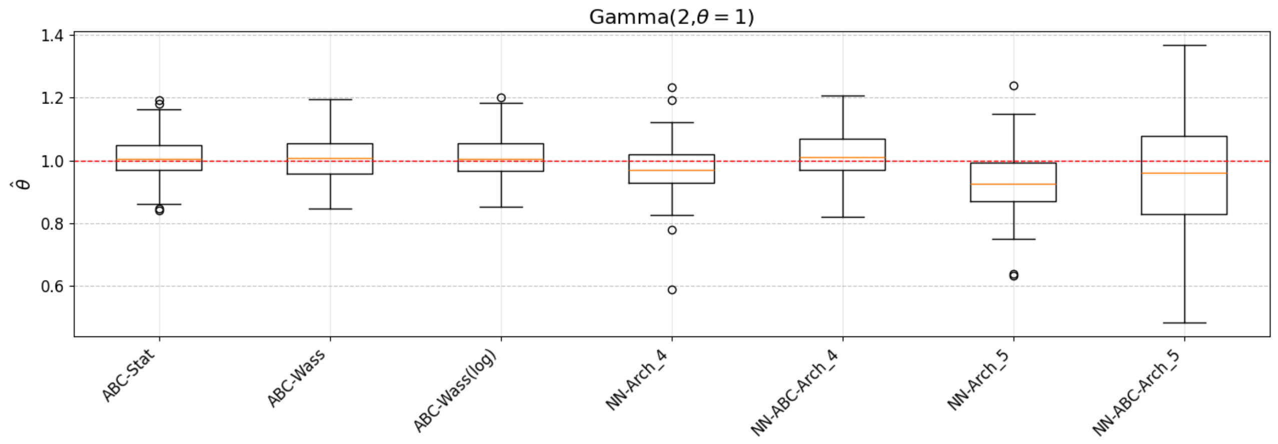


Figure C.4: Estimated posterior means for θ across 100 independent trials (true value $\theta = 1$)

D Queuing System

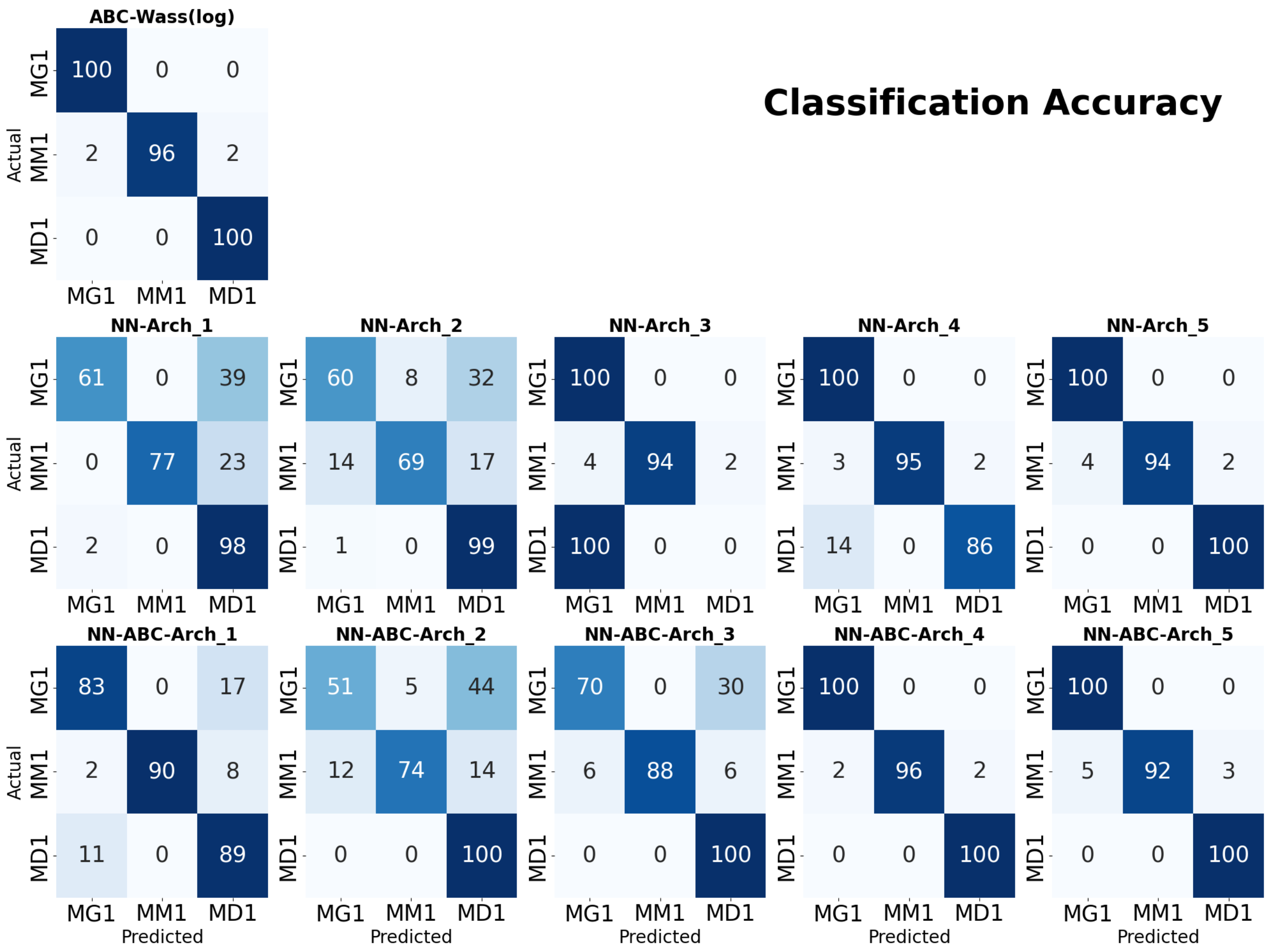


Figure D.1: Classification Accuracy: Queuing System

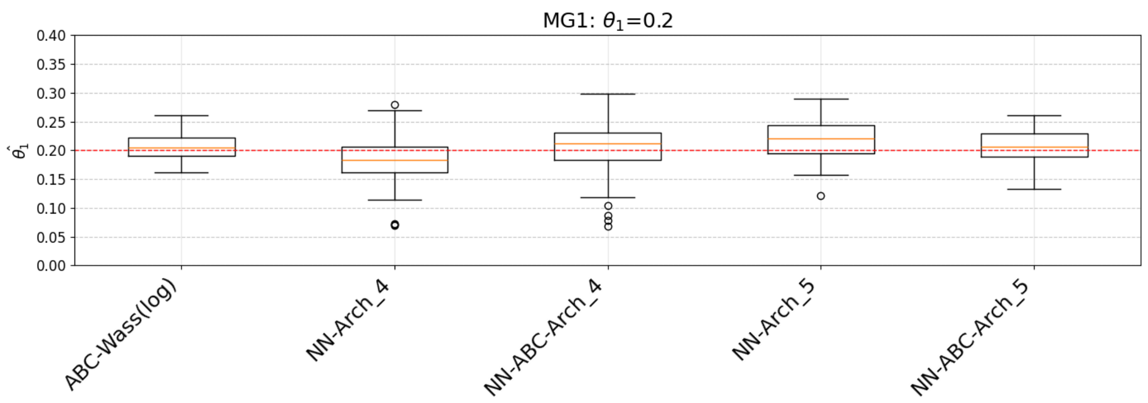


Figure D.2: Estimated posterior means for $M/G/1$: θ_1 across 100 independent trials (true value $\theta_1 = 0.2$)

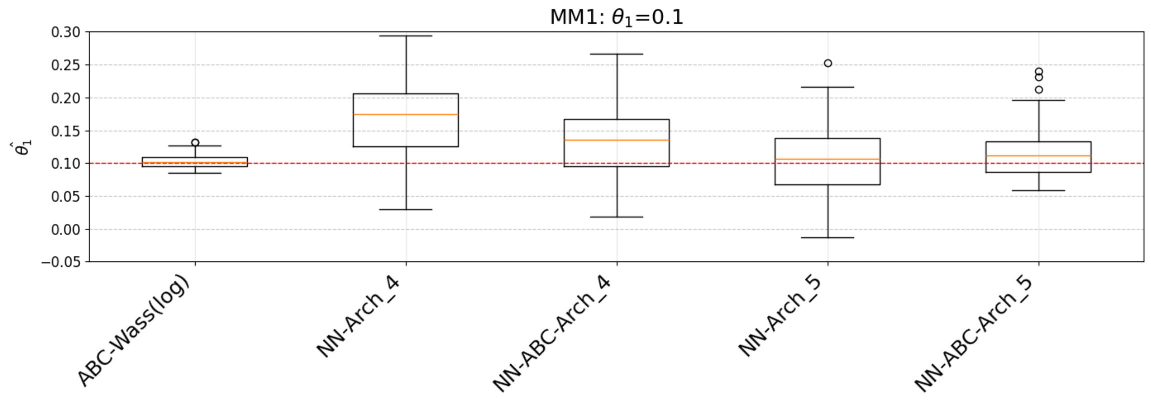


Figure D.3: Estimated posterior means for $M/M/1$: θ_1 across 100 independent trials (true value $\theta_1 = 0.1$)

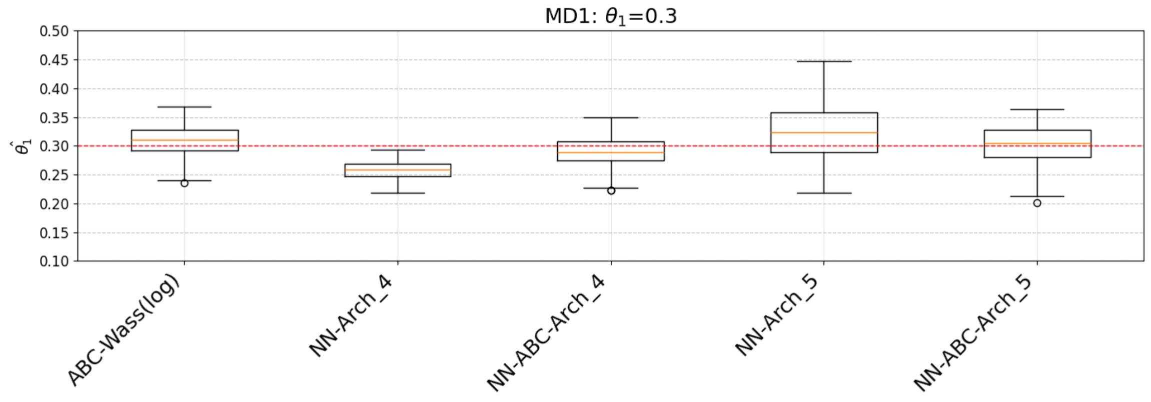


Figure D.4: Estimated posterior means for $M/D/1$: θ_1 across 100 independent trials (true value $\theta_1 = 0.3$)

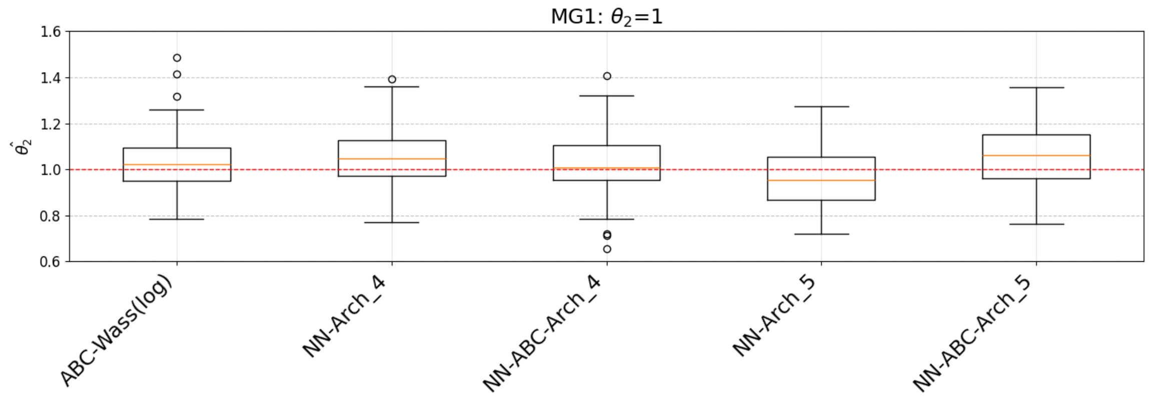


Figure D.5: Estimated posterior means for $M/G/1$: θ_2 across 100 independent trials (true value $\theta_2 = 1$)

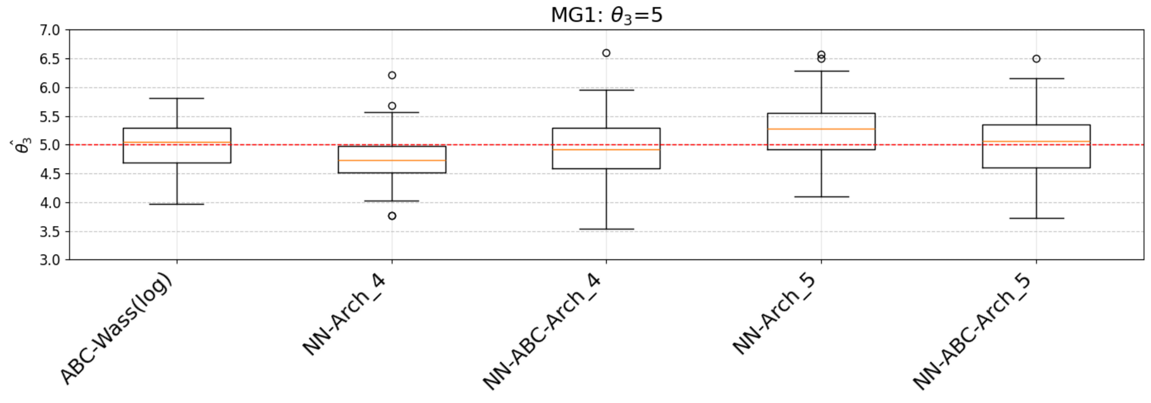


Figure D.6: Estimated posterior means for $M/G/1$: θ_3 across 100 independent trials (true value $\theta_3 = 5$)

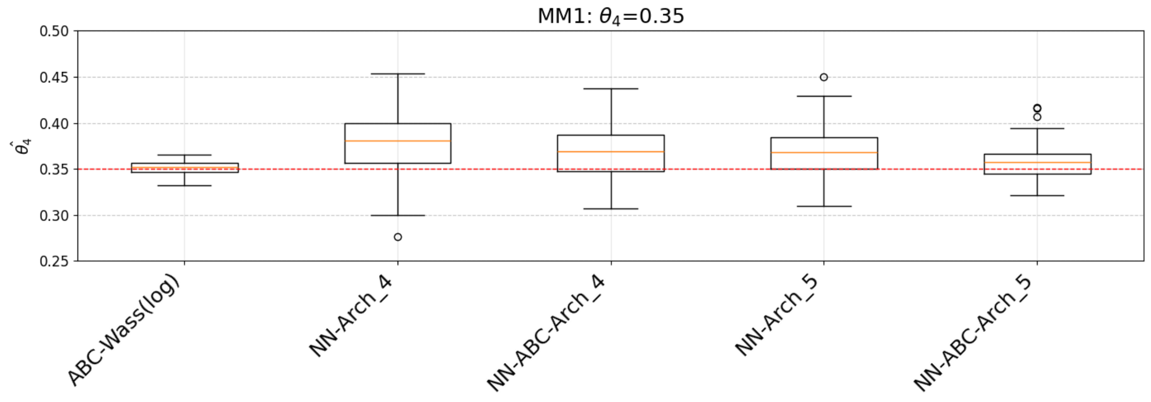


Figure D.7: Estimated posterior means for $M/M/1$: θ_4 across 100 independent trials (true value $\theta_4 = 0.35$)

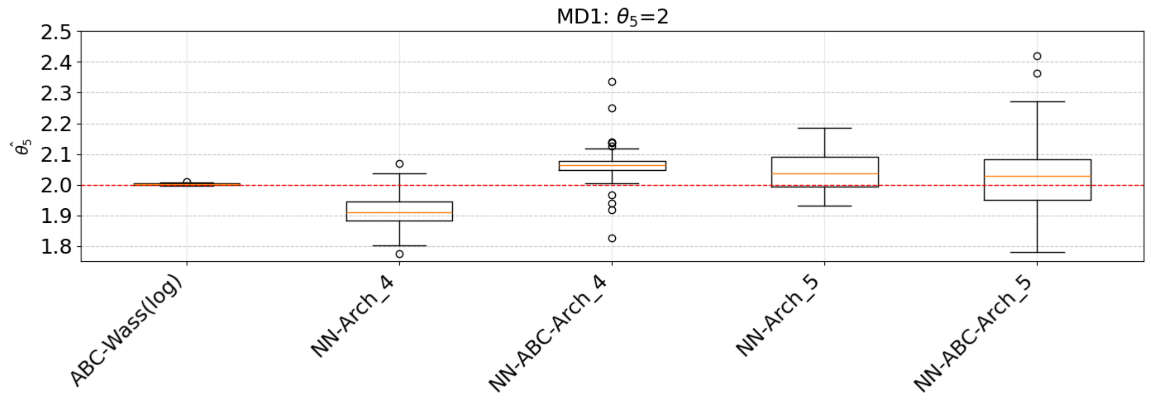


Figure D.8: Estimated posterior means for $M/D/1$: θ_5 across 100 independent trials (true value $\theta_5 = 2$)

References

Espen Bernton, Pierre E. Jacob, Mathieu Gerber, and Christian P. Robert. Approximate bayesian computation with the wasserstein distance. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 81 (2):235–269, February 2019. ISSN 1467-9868. doi: 10.1111/rssb.12312. URL http://dx.doi.org/10.1111/rssb.12312.

Gérard Biau, Frédéric Cérou, and Arnaud Guyader. New insights into approximate bayesian computation, 2013. URL https://arxiv.org/abs/1207.6461.

Jordi Burés and Igor Larrosa. Organic reaction mechanism classification using machine learning. *Nature*, 613(7945):689–695, 2023. doi: 10.1038/s41586-022-05639-4. URL https://doi.org/10.1038/s41586-022-05639-4.

Christopher Drovandi and David T. Frazier. A comparison of likelihood-free methods with and without summary statistics. *Statistics and Computing*, 32(3):42, 2022. doi: 10.1007/s11222-022-10092-4. URL https://doi.org/10.1007/s11222-022-10092-4.

Clara Grazian. Approximate bayesian computation with statistical distances for model selection, 2025. URL https://arxiv.org/abs/2410.21603.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. doi: 10.1073/pnas.79.8.2554. PMID: 6953413.

Jean-Michel Marin, Pierre Pudlo, Arnaud Estoup, and Christian P. Robert. Likelihood-free model choice, 2016. URL https://arxiv.org/abs/1503.07689.

Madhumita Mishra, T. L. Sharath Kumar, and U. M. Ashwinkumar. A novel approach to visualize arrhythmia classification using 1d cnn. In Prakash Pareek, Nishu Gupta, and M. J. C. S. Reis, editors, *Cognitive Computing and Cyber Physical Systems*, pages 198–209, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-48888-7. doi: 10.1007/978-3-031-48888-7_17.

Mijung Park, Wittawat Jitkrittum, and Dino Sejdinovic. K2-abc: Approximate bayesian computation with kernel embeddings, 2015. URL https://arxiv.org/abs/1502.02558.

Christian P. Robert, Jean-Marie Cornuet, Jean-Michel Marin, and Natesh S. Pillai. Lack of confidence in approximate Bayesian computation model choice. *Proceedings of the National Academy of Sciences*, 108(37): 15112–15117, 2011. doi: 10.1073/pnas.1102900108. URL https://www.pnas.org/doi/abs/10.1073/pnas.1102900108.

Sara Sheehan and Yun S. Song. Deep learning for population genetic inference. *PLOS Computational Biology*, 12(3):1–28, 03 2016. doi: 10.1371/journal.pcbi.1004845. URL https://doi.org/10.1371/journal.pcbi.1004845.

Siddharth Sodagi, Kanhaiya Chatla, and Siddharth Hariharan. Deep learning for ecg-based arrhythmia classification: A 1d-cnn with optimization techniques. In Mukesh Patil, Vishwesh Vyawahare, and Gajanan Birajdar, editors, *Intelligent Computing and Big Data Analytics*, pages 237–251, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-74682-6. doi: 10.1007/978-3-031-74682-6_16.