University College London

Masters Thesis for MSc in Computational Statistics and Machine Learning

An empirical study of using neural networks for conformal prediction on non-exchangeable datasets

Author
Jonathan Smith

Supervisor
Prof. John Shawe-Taylor

September 2017

1 Abstract

Neural networks are a widely used type of model for analysing data to find structure and patterns that can then be used in prediction tasks. Neural networks can be used to construct a conformal predictor, which produces predictive confidence intervals alongside its predictions.

If the underlying training and test data is generated from a probability distribution that is jointly exchangeable, these confidence intervals are provably valid, making conformal prediction a promising technique for use in safety-critical applications where the quantification of risk is required.

However, when faced with a real dataset it is never known whether the underlying distribution is exchangeable, which undermines the theoretical validity result. Despite this, it has been observed that in some applications conformal predictors achieve valid performance even when the exchangeability assumption is clearly violated.

The aim of this work was to search for indications visible during the training of neural networks that the resulting conformal predictor was likely to produce valid confidence predictions for previously unseen test data, and to use these indications to increase the likelihood of valid performance when the exchangeability assumption does not hold.

Artificial time-series datasets that violate the exchangeability assumption were used to train and test conformal predictors based on neural networks. A theoretically-motivated exploration was made of the relationship between the randomness of p-values generated by a conformal transducer on the training data and the empirical validity of the predictor's performance on unseen test data. Insight from this relationship was then integrated into the neural network training process in a novel way, in order to give an incentive to the neural network to produce valid confidence predictions.

Signs of a link were observed between the randomness of the generated p-values and the empirical validity of the performance on the test data. Overfitting was observed to have a large impact on the validity, in addition to the accuracy, of the test predictions. When the neural networks were incentivised to increase the randomness of the training data p-values, the performance of simpler conformal predictors did not change. The conformal predictors based on the more complex neural networks, however, showed a large increase in the robustness of the validity of their confidence predictions on test data in the face of overfitting.

For accompanying code see: `https://github.com/jonathan-smith-1/CSML-Thesis`

Contents

2 Introduction

2.1 The general task of prediction

Models such as neural networks are commonly used in the field of machine learning to find patterns and structure in data. Such models are often fitted or trained on existing datasets and then used to reason about future or unknown scenarios. For example, models trained on historic weather information are used to forecast tomorrow's weather, and models trained on a large number of patient medical histories could be used to provide a diagnosis for a new patient.

Whenever a model is trained on the past but used to predict future events, an assumption is made that the past is somehow representative of the future. This assumption may be implicit, or may be stated explicitly, for example that the past *training data* and the future *test data* are both comprised of independent and identically distributed (iid) values drawn from a fixed but unknown probability distribution. In this case, analysis of sufficient training data from the past allows a suitable model to characterise this distribution, and so make predictions about the future.

Models used for prediction output either *point predictions* or *hedged predictions*. Point predictions take the form of a single number (in the regression setting) or a single class (in the classification setting), for example a weather forecast's prediction that tomorrow's temperature will be 20 degrees Celsius.

Hedged predictions provide additional information regarding the confidence the model has in its own prediction. The weather forecast stating that there will be an 80% chance of rain tomorrow is an example of a hedged prediction, because it is both making a prediction (rain) and giving a statement of confidence about that prediction. The goal of any hedged predictor is to produce accurate predictions together with useful confidence information.

By quantifying the probability of an incorrect prediction, hedged predictors are suited to business-critical and safety-critical applications, where they integrate easily with risk-based compliance regimes of the sort used across many domains, including the automotive and nuclear industries [1, 2] and finance [3].

This work focuses on the *conformal predictor*, a type of hedged predictor that can be constructed using any point predictor, including neural networks. Conformal predictors produce a prediction together with hedging information in the form of statistical confidence intervals associated with the prediction.

2.2 The problem with real-world data

Some prediction techniques have theoretical bounds on their predictive performance. Initial probabilistic or distributional assumptions are first converted into mathematical statements, and then combined with a predictor and manipulated to produce these bounds. For example, for conformal predictors the assumption that the training and test data are jointly *probabilistically exchangeable*, meaning that the data was equally likely to have arrived in any order, leads directly to the proof that the predictor's confidence intervals on the test data are *valid* [4]. Being valid means that the 95% confidence interval for the next prediction contains the true value 95% of the time, and so on. Other fields, such as Bayesian statistics

and statistical learning theory each make different prior assumptions, leading to different types of performance bound at test time.

The assumptions required for conformal prediction's theoretical results are lighter than for many other prediction methods. No distributional assumptions are made, and the data is only required to be exchangeable, which can be thought of as a relaxation of the stronger and widely-used iid assumption.

The relative lightness of the single required assumption is due to the lineage of conformal prediction, which can be traced back to Kolmogorov's notion of randomness [5] and the philosophy that the core relationship between the past and the future is that the same 'style' of randomness tends to run through both.

In reality, the data scientist or engineer is presented with data whose generation method is not fully known, meaning that any assumptions cannot be rigorously checked and so theoretical bounds cannot be shown to hold. Despite this, the existence of a theoretical bound under controlled conditions is often used to informally justify model choice. However, for safety-critical and mission-critical systems this informal approach is typically not acceptable, which is a barrier to the more widespread adoption of all but the simplest machine learning models in these important systems.

2.3 Motivation for this work

This work was motivated by a study in [6] that used conformal prediction on internet network traffic demand, a time series dataset that contained long-term dependencies, so was clearly not exchangeable. Despite this, the study achieved empirically valid results on previously unseen test data, leading the authors to conclude that

'Even though the theoretical guarantee on the performance of conformal predictors is lost because of the violation of the assumption of exchangeability, the experiments showed that conformal predictors applied to time series analysis can still result in efficient and empirically valid prediction intervals.'

This naturally leads to the question of *why* empirically valid results were possible without the support of the exchangeability assumption. Furthermore, it suggests an approach for investigating this question by framing it as a model selection problem, for which a large set of techniques already exists. As shown below in Figure 1, given some data, there is a (possibly empty) set of models that lead to empirically valid predictions, and a (possibly empty) set of models that do not. The challenge is to develop a model selection technique that finds the set of models that give empirically valid performance, if such models exist.

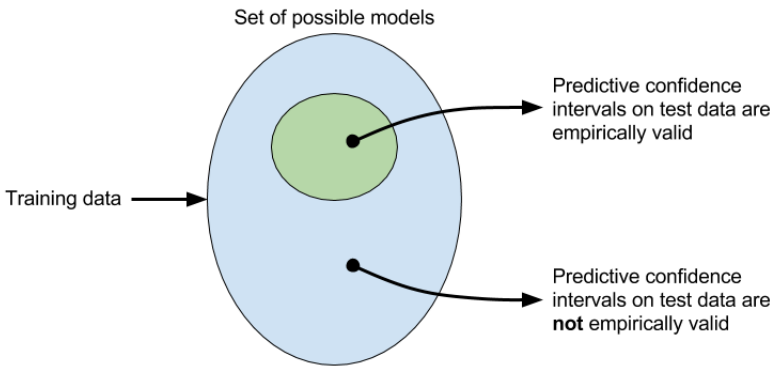


Figure 1

2.4 Application to neural networks

This work focuses on neural networks, a diverse family of flexible models that is used across all areas of machine learning and achieves current state-of-the-art performance in fields such as machine vision [7] and machine translation [8].

A neural network contains many tunable parameters, which define the set of possible models the network can form. The model fitting process consists of choosing values for these parameters and typically takes the form of a sequence of incremental updates referred to as training. The training process can be thought of as a path through the set of models in Figure 1.

2.5 Research question

The research question for this work is:

> *Given a dataset that is not probabilistically exchangeable, are there techniques to train neural networks in order to form conformal predictors that are likely to produce empirically valid confidence predictions on future test data?*

As far as it is known this work is the first to use a model selection approach with conformal prediction to target empirically valid results while deliberately violating the exchangeability assumption.

2.6 Approach, structure and aims of this work

This work attempts to answer the research question by generating artificial datasets that do not meet the exchangeability assumption, and investigating the empirical performance of a range of neural network-based conformal predictors on this data.

The goal is that insights from this empirical study provide ideas and direction for future theoretical work towards relaxing the exchangeability assumption while maintaining the performance bounds of conformal predictors, enabling a more widespread and mathematically-principled use of conformal prediction across many application domains.

2.6.1 Dataset choice

To keep the investigation close to reality, one of the datasets used in this work is generated to simulate the hourly ambulance demand in a town or city. Prediction of ambulance demand helps ambulance services prepare for surges in demand and so best deliver coverage across their community. However, incorrect demand predictions could have the opposite effect, leading to insufficient ambulance or crew availability at times of high demand, with serious consequences.

As a flexible and hedged predictor, a conformal predictor based on a neural network seems well suited to this prediction task, because the confidence intervals it produces form a transparent statement of confidence to any regulatory body or human supervisor. However, ambulance demand data contains long-term trends and is not exchangeable, so conformal prediction's theoretical validity result does not hold.

Answering this work's research question would help deliver empirically valid confidence intervals for ambulance demand prediction, and so be of value to ambulance services worldwide. It should also be noted that, while this particular dataset is inspired by ambulance demand, it does not have any ambulance-specific aspect to it that prevents the results being applied to other domains.

2.6.2 Conformal prediction

Section 5 introduces the conformal predictor and the *conformal transducer*, a closely-related construct used throughout this work. The process by which a conformal transducer converts training and test data into sets of statistical *p-values* and the corresponding meaning of validity is explained. Finally, the distinct notions of randomness for training and test data p-values are presented, together with an explanation of their relevance to this work.

2.6.3 Experiments

The investigation itself is divided into four experiments, each building on the last and increasing in sophistication. Section 6.3 explains the workflow and methodology common to all experiments, namely the data generation concept and preprocessing methods, diagnosis of how well each model fits the training data and evaluation of empirical validity on the test data.

Experiment 1 (Section 6.4) validates the new methods presented in this work for estimating the training data p-value randomness and test data empirical validity. Artificial data is generated that complies with the assumption of exchangeability, and so is known to produce randomly permuted training p-values (Section 5.6) and empirically valid performance on test data (Section 5.5). The estimate of training data p-value randomness used in this report ($p_{dev,x}$, Section 6.3.4) is shown to produce results consistent with a simulation using randomly permuted p-values. The estimate of the model's empirical validity on the test data (the 'validity gap', Section 6.3.5) is shown to give a very low score, consistent with the data.

Experiment 2 (Section 6.5) starts to investigate the relationship between training data p-value randomness and test data empirical validity. Two datasets

are generated that both have long-term trends and so do not comply with the exchangeability assumption. A spectrum of linear regression models is automatically produced, ranging from poorly-fitting to well-fitting. The better-fitting models are observed to have both a higher level of training data p-value randomness and a higher level of test data empirical validity, giving the first suggestion that these quantities are related in some way. However, this trend does not extend to the best-fitting models, which do not give correspondingly the highest level of test data empirical validity. This seems to be a similar phenomenon to overfitting, but for empirical validity, and is occurring on simple regression models that are not capable of overfitting in the usual way.

Experiment 3 (Section 6.6) moves the investigation to neural networks. It generates two datasets that have long-term trends, and trains a set of neural networks of varying complexity on the data using the backpropagation algorithm and a conventional mean square error (MSE) loss function. A modified version of the estimate of the p-value randomness is developed (the 'randomness loss', Section 6.6.2), which is a smoothed function that integrates more closely with neural networks. The randomness loss and the test data empirical validity are observed during the training process of each neural network. In the majority of cases, the point during training that achieves the highest level of test data empirical validity is close to the point that achieves the highest level of training data p-value randomness. The more complex neural networks show a tendency to overfit and, during this overfitting phase, show a highly volatile randomness loss and reduction in test data empirical validity.

Experiment 4 (Section 6.7) is a mirror of Experiment 3, using the same datasets and neural network model architectures. The difference is that the neural networks are trained using a loss function that is modified to include the randomness loss observed in Experiment 3. The randomness loss and the test data empirical validity are monitored during the training process and compared with the same quantities observed in Experiment 3 to isolate the impact of the augmented loss function. With this added incentivisation structure, the majority of models show a large reduction in randomness loss. For the more complex models seen to overfit in Experiment 3, the overfitting is reduced and the empirical validity is observed to be much more robust in the face of this overfitting, in some cases strikingly so.

3 Literature Review

3.1 Performance bounds across the field of statistics

3.1.1 Statistical learning theory

Statistical learning theory (SLT) [9] gives a theoretical foundation to many machine learning algorithms. In its usual setup the data is assumed to be iid, and drawn from an unknown distribution. The concept of *risk* is developed to measure the quality of a model [10]. Because the data's probability distribution is unknown, the risk is not computable but, with enough training data, can be approximated by the *empirical risk*. The task of model fitting is framed as restricting the possible space of models, and finding the model within the restricted space that minimises the empirical risk. Bounds can be calculated that show that the model achieves more and more accuracy with greater and greater probability as the training set grows larger. While this may provide useful information on the general quality of the model, it is an *a-priori* analysis and so not hedged prediction. It also can require a very large number of examples to produce informative results [11].

For margin-based classifiers the approach has been extended to produce performance bounds both after training [12] and at test time [13], which is a form of hedged prediction. A distributional assumption is not required, but instead the much more practical assumption is made that the regions of interest of the dataset space are 'well sampled'.

The idea of model selection as finding the best model within a restricted space influenced this work's approach of framing the search for empirical validity in terms of sets of models.

SLT naturally links to the question of whether some models are innately superior to others, across all datasets. When no assumptions are made it would appear not, as the No Free Lunch (NFL) theorem [14] states that on average across all distributions of data, all models achieve the same test error. This shows the fundamental need to incorporate constraints, restrictions or assumptions into model selection and fitting processes, and implies that the results from an empirical study such as this work may be insightful but will never be universally applicable without being backed up by theory.

The NFL theorem leads to the follow-up question of whether, once a model has been selected, theoretical performance bounds can be derived for that particular model in the total absence of assumptions. In general, again, it would appear not, which can be seen by the following reasoning on the binary classification task of predicting labels from objects in a noise-free setting. Given any fitted classifier model, consider the task of predicting the label, y_n, of a previously unseen test object, x_n:

- The classifier makes a fixed prediction, $\hat{y}_n \in \{-1, 1\}$.
- y_n has two possible values. For every distribution of the data that has $y_n = \hat{y}_n$, there is also a distribution that has $y_n \neq \hat{y}_n$.
- So averaged across all distributions, the classifier performance will be 50%, no better than chance.

This reasoning shows the intuition behind how SLT's methods of restriction of the model space, prior preferences among the models, or sufficient coverage of particular regions of the data space can give an advantage in the model selection task and produce bounds on prediction performance that are better than chance.

3.1.2 Classical statistical modelling

The classical statistical modelling approach [15] posits a parameterised model with some unknown but true values of the parameters. The data is assumed to be conditionally independent given the model and the parameters. The first task is to infer the likely model parameter values, for which confidence intervals are often produced.

This information about the parameter values can be used in a second step, called 'Posterior Fiducial Inference' by Ronald Fisher in 1935 [16]. Using this method, confidence intervals for predictions of the test data can be calculated, and under the iid assumption these confidence intervals are theoretically valid.

There is a conceptual overlap with conformal prediction, particularly in how successive errors in the online setting being iid lead to valid predictive confidence intervals [17]. However, the key difference is that the classical approach does require a distributional assumption – for example, additive iid Gaussian noise, which leads to a confidence interval defined by Student's t-distribution.

Because classical statistics is able to make use of distributional information, provided this information is correct then it is likely to produce smaller, more informative, confidence intervals than conformal prediction. These assumptions are usually investigated using diagnostic methods after model fitting. For example, if the data is assumed to have additive Gaussian noise, then the model's prediction errors should follow a Gaussian distribution and, if they do not, this suggests the original assumption is wrong. Any major structure in the prediction errors indicates a deficiency in the assumptions, and (appropriate) randomness in the prediction errors indicates that there is no more structure to be incorporated into the model.

This insight influenced this work's approach of using a conformal transducer to calculate p-values of the training data, and evaluating their randomness (Section 6.3.4). The training data p-value randomness can be thought of as a model diagnostic. If the training data p-values are found to be highly non-random, this suggests that the underlying model has not captured the structure of the data.

A philosophical extension of this idea was behind this work's approach to working with real data. Data can be thought of as a combination of structured information and (random) noise. If there exists a deterministic way to extract randomness from data, then perhaps what remains is a more succinct version of the data's structured information? In a way, this can be thought of as borrowing from Kolmogorov's notion of 'learning as a form of compression', as explained in [18], and applying it to model diagnosis as a means of checking assumptions when working with real data.

3.1.3 Bayesian methods

Bayesian methods [19] take the opposite approach to classical statistics; a family of distributional models is posited with parameters that are assumed to be random variables. In addition, information can be incorporated into prior distributions of the parameters.

Using Bayesian inference a full probability distribution of the test data can be derived, perhaps treating any unknown model parameters as latent variables. If this can be achieved then it is an extremely powerful result, because a full probability distribution contains all the probabilistic information about test prediction. For example, excluding the top and bottom 2.5% regions produces what is known as a 95% credibility interval of the value of the next test datapoint.

However, the calculation does depend on specifying the correct prior distributions and model family. In the case where the priors are incorrect, the credibility intervals may remain specific but become misleading [20]. Extensions such as robust Bayesian methods [21] may help mitigate this problem by ensuring that the model is not overly sensitive to the prior information.

3.1.4 Learning from experts

The approaches above have all encapsulated our knowledge about the future in probabilistic or distributional assumptions. Techniques involving learning from experts take a different approach: they outsource the task of predicting the future to a group of (possibly algorithmic) experts. The experts' predictions are weighted and combined to produce a point prediction. By observing the experts' predictions and the true values, experts that tend to be correct are trusted more and given a high weighting and experts that tend to be incorrect are trusted less and given a low weighting.

It is possible to bound the number of errors in a classification task to be close to the errors made by the best expert. This is very general, in that no iid or exchangeability assumption is required. By constantly adapting the weightings, performance can be maintained even when the underlying process generating the data is non-stationary [22].

In a sense, the need for assumptions on the data has passed to the experts. If an expert makes accurate predictions for long enough to receive a high weighting, that expert has managed to characterise the data and, in doing so, may have made some implicit assumptions. In any case, there is still the basic need for the past to be representative of the future, otherwise no combination of experts could hope to do better than chance in the long term.

Given that a conformal predictor can be built on top of any underlying point-prediction algorithm, prediction with experts can be used with conformal prediction to give the properties of both. Such a model created with a set of algorithmic experts was the approach used by [6], giving empirically valid performance and forming the motivation for this work as mentioned in Section 2.3.

3.2 Hedged predictions with neural networks

Other than conformal prediction, the topic of producing hedged predictions with neural networks has been approached in several ways, for which a comprehensive review can be found in [23]. One approach, known as the delta method, is to treat a neural network as a form of non-linear regression, and linearise the model in the area of interest [24, 25]. Once this approximation has been made, asymptotic theory can be used to calculate prediction intervals. A distributional assumption on the data must be made, although not necessarily Gaussian.

A second method is to produce an ensemble of neural networks and train each one on samples, drawn with replacement, of the original dataset. This produces an empirical distribution of the model's prediction [26]. For this method the predictions are assumed to have a Gaussian distribution, which the author acknowledges is for convenience rather than a fundamental requirement. In practice this was found to be problematic when applied to electricity price forecasting [27].

Finally, there are Bayesian neural networks [28], which are able to give posterior distributions of predictions but, as with all Bayesian methods, require prior distributional assumptions.

3.3 Previous applications of conformal prediction

Conformal prediction has been used successfully in several areas, almost all of which put value on the explicit quantification of risk.

3.3.1 Prediction of chemical properties from molecular structure

Conformal prediction has been combined with the random forests algorithm in a technique known as aggregated conformal prediction [29] and used in a set of related works to produce Quantitative Structure–Activity Relationship (QSAR) models, which predict a chemical's properties from its molecular structure. The ability of conformal prediction to give hedged predictions is seen as a key step to achieving regulatory approval for use of these models to assess the safety of chemicals [30, 31]. The iid assumption is made in these cases, which is a standard assumption for other QSAR models. One advantage conformal prediction appears to have over other methods in this application is that a variant known as Mondrian conformal prediction naturally manages imbalanced datasets often found in this field [32].

3.3.2 Health prognostics and diagnostics

There are many healthcare applications that are ideally suited to the ability of conformal prediction to give specific confidence information on prognostic or diagnostic information, which can be integrated with existing healthcare systems and can be easily understood by patients. Conformal prediction has been used in the diagnosis of ovarian and breast cancer [33, 34] and of the causes of acute abdominal pain [35]. Prognostic applications include psychiatric prediction tools based on neuroimaging [36] and prediction of the risk of suffering a stroke [37].

3.3.3 Nuclear fusion experiments

Experimental nuclear fusion reactors use magnetic fields to maintain a torus of plasma at 100M Kelvin. The internal temperature of the plasma cannot be measured but instead is inferred using external sensors. Conformal prediction has shown promise in the robust early detection of disruptive behaviour in the plasma, giving time for thermal quench before there is a catastrophic loss of plasma control [38].

3.3.4 Tea classification

At perhaps the other end of the spectrum there has been a surprising amount of use of conformal prediction in 'electronic noses' used for the classification of different types of tea, both in general [39] and more specifically within the Ginseng family [40].

4 Nomenclature

Sections 5 and 6 use several related terms to describe and discuss the randomness of p-values generated by the training data and the empirical validity of models considering the test data. These terms, shown in bold, are summarised below.

Training data p-value randomness is the qualitative term used throughout this report that refers to the degree of randomness of the training data.

$\boldsymbol{p_{dev,3}}$, $\boldsymbol{p_{dev,5}}$, $\boldsymbol{p_{dev,7}}$ and $\boldsymbol{p_{dev,9}}$ are quantitative estimates of the training data p-value randomness and are used in Experiments 1 and 2. They are all in the range [0, 0.5] and their method of calculation is described in Section 6.3.4. $\boldsymbol{p_{dev,x}}$ is often used to refer to all four, where $x \in \{3, 5, 7, 9\}$.

- 'Random' training data p-values will give $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,7}$ values close to 0.
- 'Non-random' training data p-values will give $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,7}$ values close to 0.5.

Randomness loss is the quantitative estimate of training data p-value randomness used in Experiments 3 and 4. It is the equivalent of $p_{dev,3}$, but uses a variant of the training data p-values referred to as synthetic p-values in this report. Its method of calculation is described in Section 6.6.2.

- 'Random' training data p-values will give randomness loss values close to 0.
- 'Non-random' training data p-values will give randomness loss values close to 0.5.

Test data empirical validity is the qualitative term referring to the degree of empirical validity of a model considering the test data.

Validity gap is the quantitative estimate of a model's level of test data empirical validity and is used in all four experiments. It is in the range [0, 1] and its method of calculation is described in Section 6.3.5.

- 'Valid' models will give a value of validity gap close to 0.
- 'Invalid' models will give a value of validity gap close to 1.

5 Introduction to Conformal Prediction

In this section the smoothed conformal transducer and the conformal predictor are introduced in the regression setting. The conformal transducer's method of calculating p-values is developed into the distinct notions of randomness of test data and training data used in this report.

5.1 Notation

This section assumes there is a set of data in the regression setting comprised of n examples, $\{(x_1, y_1), ..., (x_n, y_n)\}$, with each $y_i \in \mathbb{R}$. At this stage the data is general and is not thought of as training data or test data, and the true y_i values are all assumed to be known.

5.2 Conformal transducers

To produce a smoothed conformal transducer (referred to simply as a conformal transducer for the remainder of this report, as it is the only kind of conformal transducer that is used), a point predictor is required that makes a prediction $\hat{y}_i$ for each y_i value, usually based on its x_i value. It is possible to use any point predictor, even a constant value, but unsurprisingly predictors that capture the structure of the data produce more informative results, ultimately in the form of tighter confidence intervals. Using the chosen point predictor, prediction $\hat{y}_i$ is calculated and used to produce the nonconformity score,

$$\alpha_i = |\hat{y}_i - y_i|, \tag{1}$$

associated with each example. This is not the only possible formulation of the nonconformity score, but as the set of α_i values are only used for comparison purposes among themselves, the resulting conformal transducer and predictor are not sensitive to the 'shape' of the nonconformity score, so the simple formulation in (1) is often used. Indeed, using $\alpha_i = |\hat{y}_i - y_i|^2$ in place of (1) would not change the resulting conformal transducer or predictor at all.

The set of nonconformity scores is used to calculate the p-value of the final example, according to the smoothed conformal transducer equation,

$$p_n \equiv \frac{|\{i = 1, ..., n \mid \alpha_i > \alpha_n\}| + \tau|\{i = 1, ..., n \mid \alpha_i = \alpha_n\}|}{n}, \tag{2}$$

as presented in [41]. τ is a random sample from the uniform distribution on [0, 1].

In (2) the final example's nonconformity score, α_n, is compared to all the others. If α_n is high compared to the other nonconformity scores, y_n is considered unusual (i.e. 'nonconformal') and is assigned a p-value close to 0. Conversely, if α_n is low compared to the other nonconformity scores, y_n is considered usual and is assigned a p-value close to 1.

5.3 Validity of conformal transducers

Under the assumption that the entire dataset is drawn from an exchangeable probability distribution (meaning the data was equally likely to have been drawn in any order), p_n in (2) is uniformly distributed in the interval [0, 1]. This is the central result that underpins conformal prediction and is proven in [4]. A conformal transducer with this property is called valid.

Informally, this result can be seen as follows, assuming distinct nonconformity scores:

1. Considering that τ in (2) is a uniform random variable in the range [0, 1]:
   - If example n has the highest nonconformity score, $p_n = \frac{\tau}{n}$, which is in the range $\left[0, \frac{1}{n}\right]$
   - If example n has the next highest nonconformity score, $p_n = \frac{1+\tau}{n}$, which is in the range $\left[\frac{1}{n}, \frac{2}{n}\right]$
   - (and so on, moving down through the ranking of nonconformity scores)
   - If example n has the smallest nonconformity score, $p_n = \frac{n-1+\tau}{n}$, which is in the range $\left[\frac{n-1}{n}, 1\right]$.
2. In each case p_n occupies a distinct interval of length $\frac{1}{n}$ within [0, 1].
3. Under the exchangeability assumption, example n has an equal probability of featuring anywhere in the ranking of nonconformity scores. This means that p_n has an equal probability of being in any of the n equally-sized intervals in [0, 1].
4. Whichever interval p_n is in, its value within that interval is uniformly distributed, due to the τ term in (2).
5. Together, points 3 and 4 imply that p_n is uniformly distributed in the range [0, 1].

This reasoning extends to indistinct nonconformity scores because in that case adjacent intervals are effectively merged, resulting in unevenly-sized but appropriately-weighted intervals through the range [0, 1], leading to the same conclusion.

5.4 Conformal predictors

The conceptual jump from a conformal transducer to a conformal predictor is in the makeup of the examples $\{(x_1, y_1), ..., (x_n, y_n)\}$. In a conformal predictor, examples $\{(x_1, y_1), ..., (x_{n-1}, y_{n-1})\}$ are considered to be training examples and (x_n, y_n) is the test example. Of course, in the typical regression setting x_n is

known but y_n is not, which complicates matters because y_n is needed in order to calculate α_n in (2).

Conformal prediction works around this by simply postulating a value of y_n, letting $y_n = z$, for some $z \in \mathbb{R}$. From (2) the resulting value of p_n can then be calculated, which gives an indication of how unusual that postulated value of y_n is in the context of the entire dataset.

In its purest form, every possible value of y_n is postulated and its corresponding p-value is evaluated. A given significance level, $\epsilon \in [0, 1]$, is chosen, where $1 - \epsilon$ is referred to as the confidence level. The confidence interval associated with the prediction of y_n is the set of postulated values that have p-values greater than than ϵ[1]. Put mathematically, the $1 - \epsilon$ confidence interval is

$$\{y_n \mid p_n > \epsilon\}. \tag{3}$$

Under the assumption of exchangeability across the training and test data, the interval created for confidence level $1 - \epsilon$ will contain the true value of y_n with probability $1 - \epsilon$. Conformal predictors with this property are called valid. This validity is implied by the validity of the underlying conformal transducer.

In practice, it is unfeasible to calculate the value of p_n for every possible postulated y_n, particularly if y_n is continuous. Because of this, approaches have been developed to improve the efficiency of this stage. These approaches are based on the observation that p_n does not change smoothly with α_n, but instead only changes when α_n passes the value of one of the other nonconformity scores, changing the overall ranking order of the nonconformity scores. An example of this is the efficient implementation of a conformal predictor for an underlying ridge-regression predictor, known as the ridge regression confidence machine [42].

An important conceptual point is that the predictive confidence intervals generated by conformal predictors are statistical confidence intervals as described by Neyman in [43]. They are statements about the *method* and not about the data. Strictly speaking it is only correct to make a statement of validity before any data has been seen. Shafer and Vovk describe this in [17] and approach it from a game-theoretic point of view; valid confidence intervals can form the basis of a betting strategy that should neither make nor lose money in the long term. Once some data has been observed, a more opportunistic strategy could be possible for either side, so 'all bets are off'.

5.5 Randomness of test data p-values

As described in Section 6.3.2, in this work the datasets are artificially generated. Many possible values of test data x_n and y_n are independently generated. Under the exchangeability assumption the corresponding p_n values should form independent samples from a uniform distribution on [0, 1]. Whether these samples are likely to have come from such a distribution is considered in this report to be the 'randomness' of the test data p-values and is assessed by checking a quantity called the 'validity gap', as described in Section 6.3.5. In this report this is also referred to as test data empirical validity.

[1] In the case that this does not form an interval, the convex hull is used instead.

5.6 Randomness of training data p-values

By considering only the training data $\{(x_1, y_1), ..., (x_{n-1}, y_{n-1})\}$, a p-value for each training example can also be produced, using the conformal transducer equation

$$p_j \equiv \frac{|\{i = 1, ..., n-1 \mid \alpha_i > \alpha_j\}| + \tau|\{i = 1, ..., n-1 \mid \alpha_i = \alpha_j\}|}{n-1}, \qquad (4)$$

for each $j \in 1, ..., n-1$. τ is a random sample from the uniform distribution on [0, 1], and a different value of τ is used for the calculation of each p_j. Equation (4) is simply a restatement of (2), using a dataset of size $n-1$ and extended to each example rather than just the final one. Using the same argument as that used above for the test data p-values, under the exchangeability assumption a given training data p-value (index chosen in advance) is uniformly distributed in the range [0, 1]. However, as a collection they are highly dependent. This is easy to see from (4) for distinct nonconformity scores because, due to the ranking of nonconformity scores, each p_j occupies a different interval in [0, 1] of width $\frac{1}{n-1}$, so knowledge of some of the p_j values gives knowledge about the possible values of the others.

As such, the notion of randomness of the training data p-values is different to that of the test data p-values. In this report the training data p-values are considered random if they are spaced roughly evenly through [0, 1] and have a random permutation.

Complications arising due to indistinct nonconformity scores are ignored, as only the regression setting with continuous real values is considered. In the classification setting this case should be accounted for.

Regardless of whether the nonconformity scores are distinct or not, under the exchangeability assumption the expected value of each p_j in (4) is 0.5. This can be seen by the inherent symmetry in (4) around 0.5. Other than p_j itself, half the training data p-values are expected to be greater than p_j, and the expected value of τ is 0.5.

5.7 Point prediction

The focus of conformal prediction is on the confidence intervals, and the method of point prediction, if required, is left open to the user. Typically either the original model's prediction, $\hat{y}_n$, or the postulated value that gives the highest p-value is used. The former is efficient because $\hat{y}_n$ is required anyway to calculate the nonconformity scores. The latter has the advantage of being present in every confidence interval, so avoids the (unusual) situation where the point prediction lies outside the confidence interval. Where needed, this work uses the former.

6 Experiments

6.1 Layout of this section

This report describes four experiments, referred to as Experiments 1, 2, 3 and 4. All the experiments follow the same overall pattern, so the common features in their methods and methodology are explained first, in Section 6.3. Each experiment is then documented in turn, with sections on:

- Methodology particular to the experiment.
- Analysis of the experiment's results.
- Discussion of the experiment's results.
- Conclusions, particularly where relevant to the approach used by the next experiment.

An overall conclusion and review of the research question is made in Section 7.

6.2 Methods

For each experiment computer programs were written in the Python programming language to:

- Generate artificial training and test data, using Python's pseudo-random number generator.
- Perform any data preprocessing and transformations.
- Fit models to the training data.
- Estimate training data p-value randomness.
- Evaluate validity of model performance on the test data.

Experiments 3 and 4 use the Tensorflow python software package to create and train neural network models.

6.3 Methodology common to all experiments

6.3.1 Workflow diagram

Figure 2 shows the workflow for each experiment. The processes for the parts in red are common to all experiments and are described in this section. The model fitting is unique to each experiment and is described in each experiment's methodology section.

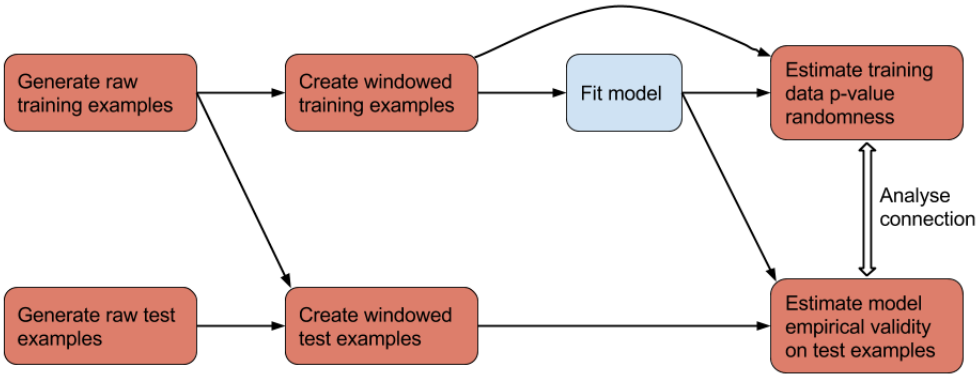


Figure 2: Experiment workflow diagram. The parts in red are common to all experiments. The model fitting is unique to each experiment

6.3.2 Training and test data generation

For each experiment 'raw' (i.e. unprocessed) training and test *examples* are generated using a data *generation rule* specific to each experiment. The rule may include a pseudo-random component and is used to generate both the training and the test data, to ensure continuity. The rule takes as input an index $i \in \{1, 2, ...\}$ and outputs object $x_i \in \mathbb{R}$ and label $y_i \in \mathbb{R}$. Each generated (x_i, y_i) is referred to as an example. Using the appropriate rule, the following data is generated for each experiment:

- A sequence of n training examples $\{(x_1, y_1), ..., (x_n, y_n)\}$, which are called the *raw training examples*. These are created using the generation rule with indices $\{1, ..., n\}$. Note that this is a slightly different notation to that introduced in Section 5.1, but is more suited to the remainder of this report.

- A set of m test examples $\{(x_{test,1}, y_{test,1}), ..., (x_{test,m}, y_{test,m})\}$, which are called the *raw test examples*. These are created using the generation rule repeatedly with index value $n + 1$. Even though they are generated using a single index value, any pseudo-random component of the generation rule may give variation among the x_{test} and y_{test} values.

Because all the test examples are generated using index value $n + 1$, they can be thought of as a cloud of possible extensions of the training sequence $\{(x_1, y_1), ..., (x_n, y_n)\}$ to the example at index $n + 1$.

An illustration of this method of data generation is shown in Figure 3 for a short sequence of data. In this example there are $n = 5$ training examples and $m = 10$ test examples. In this case the data generation rule is:

- $x_i = i$

- $y_i = x_i^2 + \epsilon_i$, where each ϵ_i is an independent sample from a normal distribution with mean 0 and standard deviation $\frac{i^2}{10}$

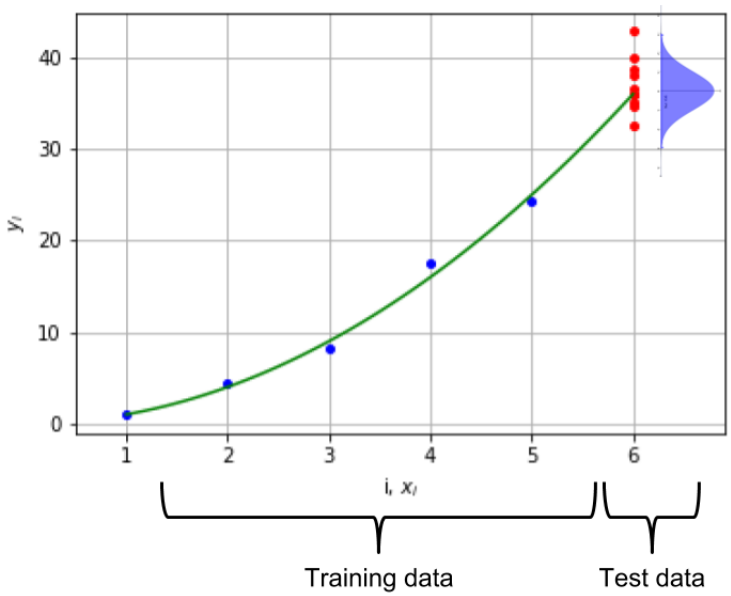


Figure 3: Training and test data generation concept for $y_i = x_i^2 + \epsilon_i$. The test y values were generated by sampling from a Gaussian distribution with mean 36 and standard deviation 3.6.

6.3.3 Training and test data preprocessing

Training data window transform

Each raw training example is combined with previous examples to create new objects. This work takes the approach described in [6] and [44], but adapts it to the regression setting. In this report this process is referred to as a *window transform*. The window transform has an associated *window length*, which is the number of raw examples that are combined to create the new, windowed, examples. In this report these new examples are called *windowed training examples* or just *training examples*.

For example, applying a window transform of length 3 to raw training examples $\{(x_1, y_1), ..., (x_n, y_n)\}$ gives the set of $n - 2$ windowed training examples,

$$\begin{aligned} \{ &([x_1, y_1, x_2, y_2, x_3], y_3), \\ &([x_2, y_2, x_3, y_3, x_4], y_4), \\ &..., \\ &([x_{n-2}, y_{n-2}, x_{n-1}, y_{n-1}, x_n], y_n)\}, \end{aligned} \tag{5}$$

where each windowed example is formed by combining 3 raw examples. In this report, by convention a window transform of length 1 leaves the raw training examples unchanged.

Window transforming is commonly used when analysing time-series data because combining raw training examples gives the model visibility of how the data evolves over time. This allows the model not only to find relationships between the y_i labels and x_i objects, but also relationships between sequential y_i labels.

Test data window transform
Each raw test example is combined with previous examples to create new objects, in a similar manner to the training examples. However, because each test example is a possible extension of the sequence of training examples, its set of previous examples consists of the final raw training examples rather than other raw test examples, as shown in Figure 3. In this report the resulting new examples are called *windowed test examples* or just *test examples*.

For example, applying a window transform of length 3 to raw training examples $\{(x_1, y_1), ..., (x_n, y_n)\}$ and raw test examples $\{(x_{test,1}, y_{test,1}), ..., (x_{test,m}, y_{test,m})\}$ gives the set of m windowed test examples,

$$\begin{aligned}
&\{ \left(\left[x_{n-1}, y_{n-1}, x_n, y_n, x_{test,1}\right], y_{test,1}\right), \\
&\ \left(\left[x_{n-1}, y_{n-1}, x_n, y_n, x_{test,2}\right], y_{test,2}\right), \\
&\ ..., \\
&\ \left(\left[x_{n-1}, y_{n-1}, x_n, y_n, x_{test,m}\right], y_{test,m}\right)\},
\end{aligned} \tag{6}$$

where each windowed test example is formed by combining a raw test example with the final 2 raw training examples. In this report, by convention a window transform of length 1 leaves the raw test examples unchanged.

6.3.4 Estimation of training data p-value randomness

Calculation of p-values on the training set
After the model has been fitted it is used to form a conformal transducer and calculate the p-values associated with each of the training examples, as described in Section 5.6. In this report these are called the *training data p-values*.

Calculation of p-value moving average mean deviation from 0.5
As described in Section 5.6, the p-values produced by a conformal transducer on an exchangeable dataset are randomly permuted, are approximately evenly spaced in the interval [0, 1] and have an expected value of 0.5. This permutation-based randomness is outside the scope of the usual test for uniform randomness used in conformal prediction, the plug-in martingale [45, 46]. Because of this, a method based on the moving-average of the p-values is developed and used in this report.

If the p-values are in a random order, then a continuous sequence of p-values will have a mean value of around 0.5. If the p-values are not in a random order, then similar p-values may be bunched together and short sections will have means that deviate from 0.5. This level of deviation of p-value means from 0.5 is used to assess whether the p-values are in a random order. Moving averages are used to calculate the means of sections of p-values.

The moving average of each set of p-values is calculated using window sizes of 3, 5, 7 and 9. Odd numbers are chosen to allow the moving average window to extend equally before and after the index position being assessed. For example, for a window size of 5 the moving average value of p_i is

$$p_{avg_5,i} = \frac{p_{i-2} + p_{i-1} + p_i + p_{i+1} + p_{i+2}}{5}, \tag{7}$$

where the indexing wraps around, so the index after n is 1 and the index before 1 is n.

For each set of averaged p-values the mean absolute deviation from 0.5,

$$p_{dev,x} = \frac{1}{n} \sum_{i=1}^{n} |p_{avg_x,i} - 0.5|, \tag{8}$$

is calculated, with subscript x standing in for any of $\{3, 5, 7, 9\}$. $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,9}$ are used and assessed as estimates of the training data p-value randomness in Experiments 1 and 2. A modified version of $p_{dev,3}$ called *randomness loss* is used in Experiments 3 and 4. Its calculation method is given in the methodology section of Experiment 3, Section 6.6.2.

6.3.5 Estimation of model empirical validity on test data

Calculation of test p-values
Using the fitted model, a conformal transducer is constructed and a separate p-value associated with each test example is calculated, as described in Section 5.2. In this report these are called the *test data p-values*.

Estimate of empirical validity
In this report the empirical validity as defined in Section 5.3 is assessed by examining the test data p-values. All significance levels are assessed simultaneously, based on the following argument:

- Consider any significance level $\epsilon \in [0, 1]$.
- A conformal predictor's $(1 - \epsilon)$ predictive confidence interval of $\hat{y}_{test}$ is the set of values[2] of $\hat{y}_{test}$ such that $p_{test} > \epsilon$.
- For a valid predictor, proportion $(1 - \epsilon)$ of the test examples is expected to be inside this interval. For example, 95% of examples are expected to be inside the 95% confidence interval.
- Therefore, for a valid predictor, proportion $(1 - \epsilon)$ of the test examples is expected to have $p_{test} > \epsilon$. For example, 95% of examples are expected to have $p_{test} > 0.05$.

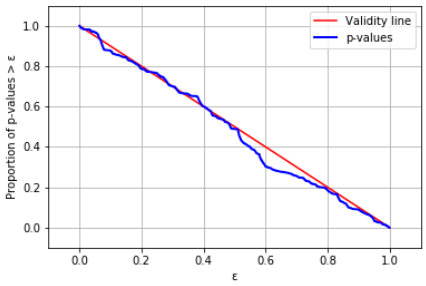
This calculation can be visualised for all ϵ values simultaneously by producing a plot such as those shown in Figure 4, with ϵ on the horizontal axis and the proportion of $p_{test} > \epsilon$ on the vertical axis[3]. In this report this type of plot is called a *validity plot*. As the proportion of p-values greater than ϵ only changes when ϵ passes through a p-value, the calculation only needs to be performed for ϵ values that are the p-values.

[2]Strictly speaking, the convex hull of the set of values.

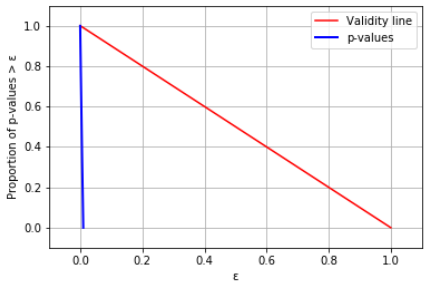
[3]By flipping the vertical axis you arrive at the empirical cumulative distribution function of the p-values, which for iid p-values uniformly distributed in [0, 1] should be a straight line through the origin with gradient 1.

In this report the red '$1-\epsilon$' line is called the *validity line*. If the plotted values are all close to the line, as in Figure 4(a), then the transducer performance is close to empirically valid across the entire [0, 1] confidence range.

To allow a quantitative comparison of empirical performance across different models and datasets, the proximity of the plotted lines to the validity lines is reduced to a single number by calculating the mean absolute vertical distance between the plotted p-values and the validity line. This gives an estimate of deviation from validity, which this report calls the *validity gap*. Validity gap values close to zero imply that the model's performance is close to empirically valid. Mean absolute distance is used rather than area between the line described by the p-values and the validity line because the test p-values may only occupy a small region of interval [0, 1] (as in Figure 4(b)), making an area calculation outside this region unclear.



(a) Example validity plot, with a low validity gap of 0.0233 consistent with the p-values being close to the validity line.



(b) Example validity plot, with a high validity gap of 0.4957 consistent with the p-values being far from the validity line.

Figure 4: Example validity plots

Estimate of prediction performance

Where required, model predictive performance on the test examples is calculated. While the focus of this report is empirical validity, predictive performance is also tracked because a model with poor predictive performance may not be useful, even if it is empirically valid.

The mean absolute prediction error,

$$\hat{y}_{err,mean} = \frac{1}{m}\sum_{j=1}^{m}|\hat{y}_{test,j} - y_{test,j}|, \tag{9}$$

is used as the measure of model predictive performance.

6.4 Experiment 1

6.4.1 Introduction

The aim of Experiment 1 was to demonstrate:

- The chosen methods of estimating the randomness of training data p-values
- The chosen method of estimating a model's empirical validity on test data

on data known to be probabilistically exchangeable, and therefore known to have random p-values and empirically valid test predictions.

6.4.2 Methodology

Raw training and test data generation
1000 training examples were generated using indices $i = 1, ..., 1000$ and 1000 test examples were generated all using index $i = 1001$. The data generation rule was:

- $x_i = i$
- All y_i are independently sampled from a standard normal distribution.

Prior to generating these samples, the seed of python's pseudo-random number generator was set to 0, for repeatability.

Model choice
The model used in Experiment 1 was a sample mean predictor, where the predicted value of each y_i was the sample mean of the training data labels,

$$\hat{y}_i = \frac{1}{1000} \sum_{i=1}^{1000} y_i. \tag{10}$$

Training data window transform
In Experiment 1 a window length of 1 was used.

Summary of Experiment 1 configuration

Test No.	Dataset type	Window Length	Model
1	Standard normal	1	Sample mean

Table 1: Experiment 1 model fitting configuration.

Experimental procedure
The experimental procedure for Experiment 1 is shown in the workflow diagram in Figure 5.

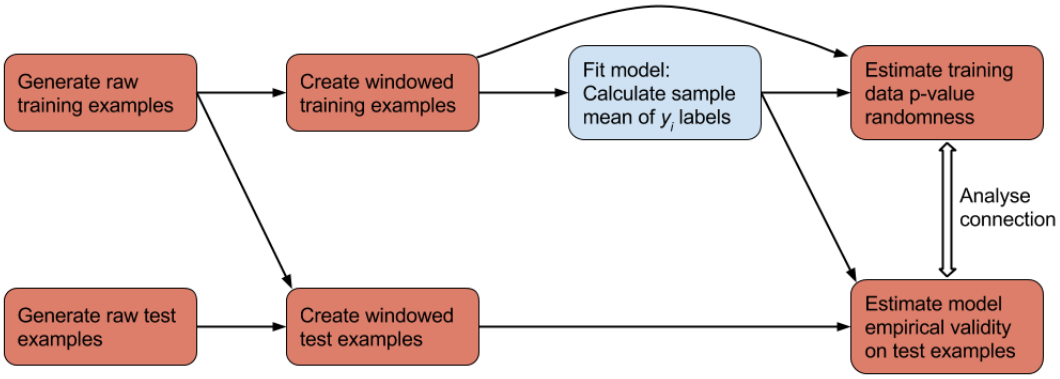


Figure 5: Experiment 1 workflow diagram. The parts in red are common to all experiments. The parts in blue are specific to this experiment.

6.4.3 Results

The results of Experiment 1 are shown in Table 2.

Test No.	Training diagnostics				Test performance	
	$p_{dev,3}$	$p_{dev,5}$	$p_{dev,7}$	$p_{dev,9}$	Validity gap	Pred. error
1	0.139	0.108	0.089	0.077	0.018	0.818

Table 2: Experiment 1 results. $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,9}$ are estimates of how far the training data p-values are from being random, as detailed in Section 6.3.4. Validity gap is an estimate of how far the test p-values are from being empirically valid, as detailed in Section 6.3.5. Pred. error is the mean absolute prediction error on the test data, also detailed in Section 6.3.5.

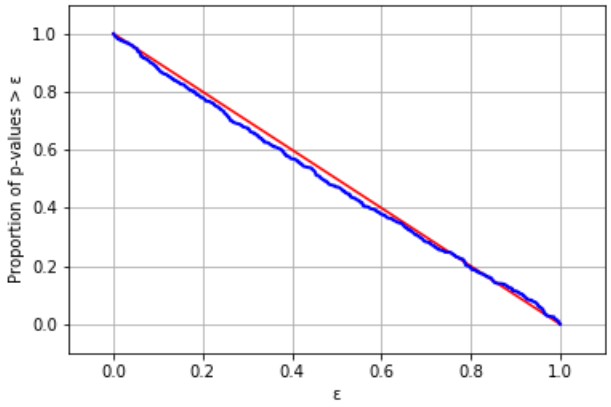


Figure 6: Experiment 1 validity plot

6.4.4 Analysis and Discussion

Model fitting diagnostics
Because the training and test data were known to be iid, and therefore exchangeable, the resulting smoothed conformal predictor was known to be theoretically valid [47]. Being exchangeable, the training and test examples were equally likely to have been generated in any order. This can be used to validate the calculated values of $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,9}$ against simulated values. Simulated values were calculated, as described in Appendix C, by generating 10,000 sets of p-values, then randomly reshuffling them and calculating their $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,9}$ values. A comparison is shown in Table 3.

	$\boldsymbol{p_{dev,3}}$	$\boldsymbol{p_{dev,5}}$	$\boldsymbol{p_{dev,7}}$	$\boldsymbol{p_{dev,9}}$
Observed	0.139	0.108	0.089	0.077
Simulated	0.134 ± 0.011	0.102 ± 0.012	0.085 ± 0.013	0.074 ± 0.013

Table 3: Experiment 1 training data p-value observed and simulated randomness results. $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,9}$ are estimates of how far the training data p-values are from being random, as detailed in Section 6.3.4. The spread of the simulated values is the root mean square (RMS) deviation from the mean across the 10,000 simulated sets of p-values.

It can be seen that the observed $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,9}$ values (created with data known to be iid) were close to the simulated values (with p-values shuffled, introducing exchangeability). $p_{dev,9}$ is smaller than $p_{dev,7}$, which in turn was smaller than $p_{dev,5}$, and so on, due to the smoothing effect of the larger window sizes.

Model empirical validity
The calculated validity gap of 0.018 was low, indicating that the model's performance was close to being empirically valid, which is consistent with the knowledge that for this iid data the model is theoretically valid. The validity gap being non-zero was due to the natural variation in randomly-generated datasets.

6.4.5 Conclusion

Using a known dataset, estimates $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,9}$ of training data p-value randomness were calculated and shown to be close to values found by simulation. The validity gap of the test p-values was found to be small, which was consistent with the known test data empirical validity of the chosen dataset.

6.5 Experiment 2

6.5.1 Introduction

The aim of Experiment 2 was to:

1. Produce a spectrum of linear regression models known to range from well-fitting to poorly-fitting.

2. Examine how estimates of training data p-value randomness and test data empirical validity change across the range of models.

The approach to producing a spectrum of models was first to produce well-fitting and poorly-fitting models, and then blend between the two using a weighted average of the coefficients, as described in Section 6.5.2. The resulting range of models varied smoothly from well-fitting to poorly-fitting.

6.5.2 Methodology

Raw training and test data generation
For Experiment 2 two types of data were generated: quadratically increasing data and simulated ambulance data.

Quadratically increasing data was chosen because it is simple data with a strong trend, and therefore is far from being probabilistically exchangeable. Furthermore, the trend is non-linear, presenting a further challenge for the model fitting. Simulated ambulance data was chosen because it has a more complex, cyclical, structure and is relevant to the real-world problem of ambulance demand prediction.

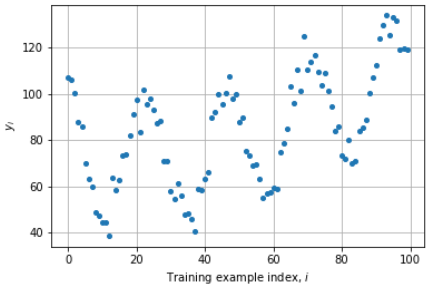
In both cases five complete (training and test) datasets were generated. These five sets were generated using the same generation rule but with the seed of python's pseudo-random number generator set to each of 0, 1, 2, 3, 4 beforehand. For each dataset 100 training examples were generated using indices $i = 1, ..., 100$ and 100 test examples were generated using index $i = 101$. A dataset of size 100 was large enough to give fine-grained confidence results, but small enough to perform the calculations for this experiment in a reasonable time.

Quadratic data generation
For the quadratic data the generation rule was:

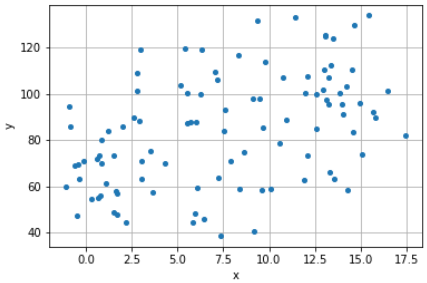
- $x_i = i$
- $y_i = x_i^2 + \epsilon_i$, where each ϵ_i was an independent sample from a standard normal distribution.

Simulated ambulance data generation
The simulated ambulance data was designed to simulate real-world hourly ambulance demand in a town or city. The i index is considered to be a time index, in hours. Each x_i was the outdoor temperature at hour i and each y_i was the number of ambulance callouts during hour i. The ambulance demand was a function of outdoor temperature and other factors such as hour i (weekly and annual cycles)

and an overall increasing trend (to simulate population growth). Details of the generation rule used for the simulated ambulance data are in Appendix B. The simulated ambulance dataset generated with seed 0 is shown in Figure 7. Figure 7(b) shows how weak the relationship was between 'outdoor temperature' and 'time', demonstrating the necessity of the window transform to allow a time-series approach to be taken.



(a) Simulated ambulance demand (y_i) against time (i), over 100 hours.



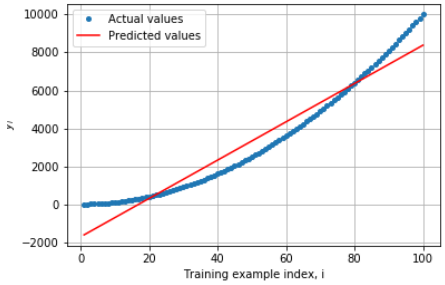
(b) Simulated ambulance demand (y) against outdoor temperature (x), over 100 hours.

Figure 7: Experiment 2 simulated ambulance data.

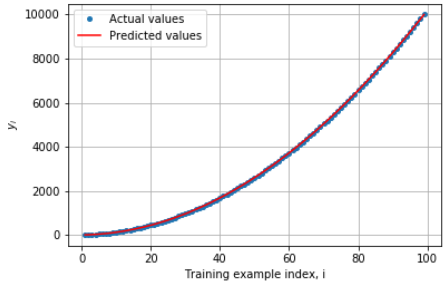
Model choice

For each of the five generated datasets (seed 0, 1, 2, 3, 4) and two dataset types (quadratic and ambulance):

- A window transform of length 1 was performed and the data was standardised to zero-mean and unit variance. A linear regression model (Model 1) was fitted. Because the x_i objects were scalar, Model 1 had two coefficients: an offset and a linear term.
- A window transform of length 2 was performed and the data was standardised to zero-mean and unit variance. A linear regression model (Model 2) was fitted. Because each x_i object had three components, Model 2 had four coefficients: an offset and three linear terms.

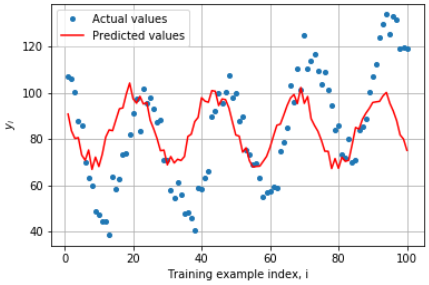


(a) Model 1, with window length 1.

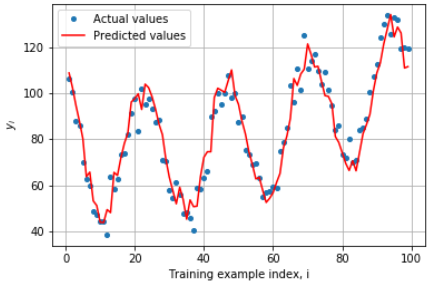


(b) Model 2, with window length 2.

Figure 8: Model 1 and Model 2 prediction performance on the quadratic training data. The datasets shown were generated using a seed of 0 and the models were fitted to this data.



(a) Model 1, with window length 1.



(b) Model 2, with window length 2.

Figure 9: Model 1 and Model 2 prediction performance on the simulated ambulance training data. The datasets shown were generated using a seed of 0 and the models were fitted to this data.

As shown in Figures 8 and 9, for both dataset types Model 1 fitted poorly and Model 2 fitted well.

Model 1 could be completely expressed in the form of Model 2 by creating two new coefficients and setting them to zero. Each of the four coefficients could then be smoothly varied from the value found by Model 1 to the value found by Model 2. To control the process, a blending factor in the range 0 to 1 was used to give a linear weighted average of the coefficients of each model. A blending factor of 0.0 reproduced Model 1, and a blending factor of 1.0 reproduced Model 2, with a linear variation of each coefficient for values in between.

A workflow diagram for Experiment 2 is shown below in Figure 10.

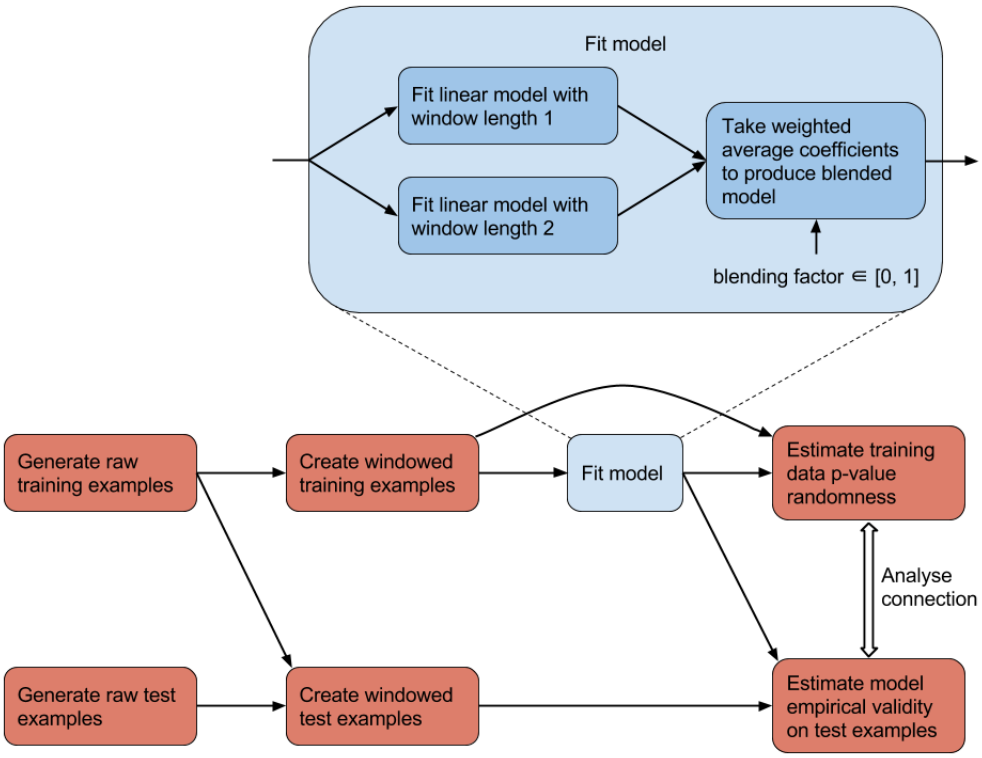


Figure 10: Experiment 2 workflow diagram. The parts in red are common to all experiments. The parts in blue are specific to this experiment.

Based on feedback from some preliminary runs, for the quadratic datasets 1000 evenly spaced blending factor values were used in the range [0.95, 1.0], as this is the range in which the model performance sees most of its variation. For the simulated ambulance datasets, 1000 evenly spaced blending factors were used in the range [0.0, 1.0].

The training data p-value randomness and the test data empirical validity were estimated and plotted. Because of the variation in blending factors, the plots represent trajectories between Model 1 and Model 2 in a $p_{dev,x}$ and validity gap 'space'.

Additionally, a window transform of length 1 followed by fitting a quadratic regression model was performed on the quadratic data and plotted. Because the data was known to be quadratic, this model matched the data and so fitted extremely well, so was used as a baseline for comparison with the linear regression models.

A summary of which models were fitted to which dataset types is shown in Table 4.

Test No.	**Dataset type**	**Window Length**	**Model**
1	Quadratic	$1 \rightarrow 2$	Linear, blended
2	Quadratic	1	Quadratic
3	Ambulance	$1 \rightarrow 2$	Linear, blended

Table 4: Experiment 2 regression model fitting configurations. $1 \rightarrow 2$ means that the model was blended between a linear regression model fitted using a window length of 1 and a linear regression model fitted using a window length of 2.

6.5.3 Results

Results for the validity gap and $p_{dev,3}$ for both datasets are presented in Figures 11 and 12 below. Full results for $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,9}$ are presented in Appendix A.1.

Quadratic data

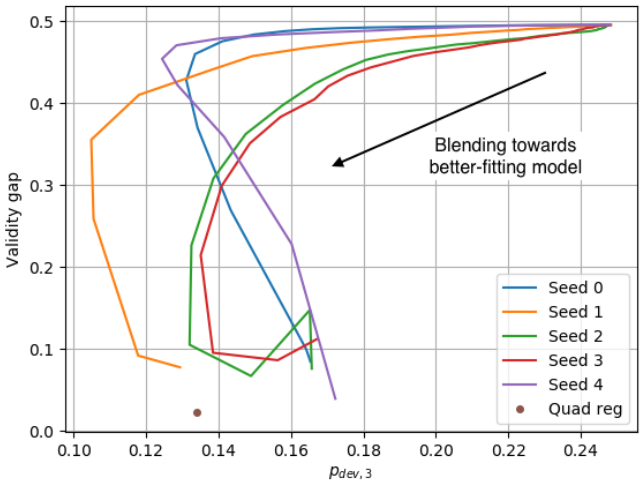


Figure 11: Trajectory through the validity gap / $p_{dev,3}$ space of linear model performance on quadratic data as model coefficients are blended from a poorly-fitting model to a well-fitting model. Five different datasets are shown, generated with different random seeds. Validity gap is an estimate of how far the test p-values are from being empirically valid, as detailed in Section 6.3.5. $p_{dev,3}$ is an estimate of how far the training data p-values are from being random, as detailed in Section 6.3.4. The point value of a quadratic regression model is also plotted.

Ambulance data

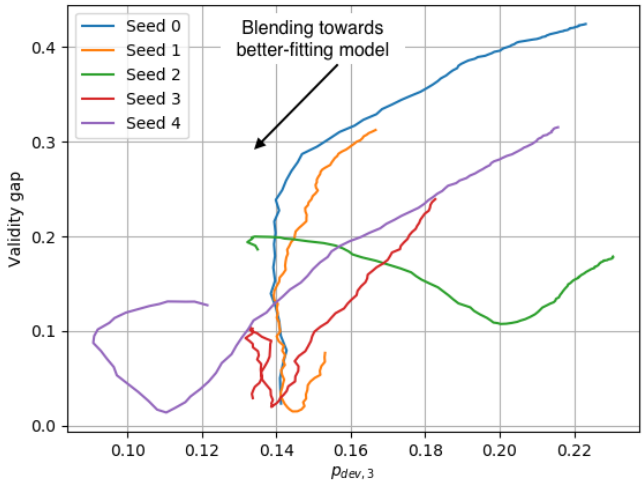


Figure 12: Trajectory through the validity gap / $p_{dev,3}$ space of linear model performance on simulated ambulance data as model coefficients are blended from a poorly-fitting model to a well-fitting model. Five different datasets are shown, generated with different random seeds. Validity gap is an estimate of how far the test p-values are from being empirically valid, as detailed in Section 6.3.5. $p_{dev,3}$ is an estimate of how far the training data p-values are from being random, as detailed in Section 6.3.4.

6.5.4 Analysis

Quadratic data

For the quadratic data the trajectories for each of the randomness estimates $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,9}$ and for each seed started from approximately the same point of $p_{dev,x} \approx 0.25$ and a validity gap ≈ 0.5. As the blending factor changed from 0.95 towards 1.0, the $p_{dev,x}$ values decreased more rapidly than the validity gap, and there was considerable variation between the trajectories. In all cases the initial direction of movement was decrease in $p_{dev,x}$ and decrease in validity gap.

As the blending factor increased further, in almost all cases the $p_{dev,x}$ scores started to increase again, with $p_{dev,3}$ and $p_{dev,5}$ showing the biggest reversal. As the blending factor increased still further, in some cases the validity gap also increased, causing the trajectory to curl around and move back towards the direction it came from. In all cases any curling pattern was counter-clockwise.

There was considerable variation between the finishing points of the trajectories, but all finished in the same region of the space. The randomness estimates with the longer window lengths ($p_{dev,7}$ and $p_{dev,9}$) had a tighter spread of finishing points and less curled trajectories than the estimates with the smaller window lengths ($p_{dev,3}$ and $p_{dev,5}$), but otherwise for each dataset there was a strong similarity between the results for the four $p_{dev,x}$ estimates.

Most of the trajectories approached the point achieved by the quadratic regression model at some point on their trajectory, but none matched its validity gap.

Simulated ambulance data

For the simulated ambulance data the trajectories started at very different points across the five datasets and across the four $p_{dev,x}$ randomness estimates. As the blending factor changed from 0.0 towards 1.0, all four $p_{dev,x}$ values and the validity gap started to decrease for all trajectories, across all datasets.

Following the initial movement, the behaviour of the trajectories started to differ. The trajectory for seed 0 saw its $p_{dev,x}$ value mostly stop changing, but its validity gap reduced to a value of around 0.03. The trajectories for seeds 1 and 4 exhibited curling behaviour, with seed 1's trajectory curling counter-clockwise and seed 2's trajectory curling clockwise. Seed 2 achieved a validity gap close to 0.1, then increased its validity gap to around 0.2 and finally performed a sharp reversal as the blending factor approached 1. Seed 3 achieved a validity gap of around 0.03 before zig-zagging up to around 0.1 and down to 0.03 again.

Overall there was a large degree of similarity in the trajectory shapes between the different $p_{dev,x}$ estimates, but a large difference across the five seeds. All the trajectories finished in very different places.

6.5.5 Discussion

Trajectory start and finish points
The trajectory start and finish points are simply the performance of Model 1 and Model 2 on the generated datasets, so the focus of this discussion is on the shape of each trajectory between these points.

Initial relationship between $p_{dev,x}$ and validity gap
Without exception the results show an initial decrease in $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$, $p_{dev,9}$ and validity gap as the model fit improves, showing that conventional model fitting also can have the effects of increasing the training data p-value randomness and reducing the validity gap. This is not surprising, as model fitting involves capturing the underlying trends in the data which, if known, should help with prediction. The fact that the $p_{dev,x}$ estimates (which are only calculated using the training data) also decrease at first during training suggests that improving model fit and reducing $p_{dev,x}$ are initially somewhat aligned, meaning that the $p_{dev,x}$ estimates could be useful quantities to help with the fitting process.

Trajectory shape and 'empirical overfitting'
The cause of the curling behaviour of the trajectories is not clear. The levelling-off and then increase in validity gap as model fitting improves is perhaps the equivalent of overfitting for empirical validity, where the model becomes overly suited to the training data and does not generalise well to the test data. This would be consistent with the validity gap showing larger increases on the simulated ambulance data than the quadratic data, because the simulated ambulance data has a more complex structure and so perhaps is more at risk from this kind of overfitting.

Overfitting in the conventional sense would seem unlikely as even the more complex Model 2 has only 4 parameters and is modelling a training set with 100 examples. However, it would seem that, depending on the dataset, validity gap can be extremely sensitive, as demonstrated by the performance on the quadratic data being almost unaffected by blending factors between 0 and 0.95 in an initial study. This is consistent with there being relatively little noise in the quadratic data: the additive noise is samples from a standard normal distribution ϵ_i for y_i values in the range 1^2 to 100^2. To be remotely empirically valid the model has to be very accurate indeed and 'hit' the central region of the distribution of test examples, which is Gaussian distributed with a mean of $101^2 = 10201$ but standard deviation of only 1.

Misalignment of $p_{dev,x}$ and training error
It is not clear why in almost all cases the $p_{dev,x}$ values drop and then start to increase again as the blending factor approaches 1.0. Given that the $p_{dev,x}$ values are only calculated on the training data, they would be expected to keep improving as the model fits better. The fact that improving a model by reducing its MSE training loss can increase its $p_{dev,x}$ values (indicating training data p-values that are less random) points to a misalignment between these two indicators of model fit. For badly fitting models, it seems that reducing the training error also reduces the $p_{dev,x}$ values, but once a model is fitting well it seems that this relationship no longer holds.

One explanation for this could be the fundamental difference between training errors and training data p-values. P-values, as calculated by a conformal transducer, are a gauge of the relative sizes of nonconformity scores. When training errors (and thus nonconformity scores) are very large, the ranking of the nonconformity scores is unlikely to change quickly, so the p-values will not change quickly. Once the training errors are all small, however, minor changes in the model can cause large changes in the ranking order of the nonconformity scores, causing large changes in the p-values.

Whatever the cause, once a model is fitting well, there does seem to be a misalignment between minimising the MSE training loss and increasing the randomness of the training data p-values. This is encouraging, because it suggests that training data p-value randomness as defined in this report is giving different information about the training process than is given by the training error.

Similarity across $p_{dev,x}$ values

It is noticeable that in all cases $p_{dev,3}$ exhibits the most curling and twisting behaviour of the trajectories. $p_{dev,5}$ shows less, $p_{dev,7}$ even less and $p_{dev,9}$ the least. This is unsurprising because the longer window lengths show less random deviation in their mean and represent larger sections of data. Overall, the similarity between all four $p_{dev,x}$ means that none of them offers any particular insight that the others do not.

6.5.6 Conclusion

On the datasets considered in Experiment 2 there seemed to be a connection between $p_{dev,x}$ values and validity gap for poorly-fitting models, and the trend was that these two quantities both decreased as the model improved. For well-fitting models the picture was less clear. Once the models fitted well, $p_{dev,x}$ seemed to behave differently to training error, so is worth examining further as a source of insight for the model fitting process, particularly as the fitting process of minimising the MSE training loss used in this experiment did not seem to fully align with achieving a low validity gap. All four $p_{dev,x}$ estimates seemed to give a similar level of insight, so only $p_{dev,3}$ was used for the remainder of this report. Finally, there was a hint of a phenomenon similar to overfitting but for empirical validity, where validity gap as evaluated on the test data worsened as the model's fit on the training data improved.

6.6 Experiment 3

6.6.1 Introduction

Experiment 3 used neural networks of varying architectural complexity as the basis for conformal transducers, and examined how the training data p-value randomness and test data empirical validity evolved as the neural networks were trained by backpropagation. A neural network with randomly initialised parameters will almost certainly be a poorly-fitting model, and a trained neural network is likely to fit much better, so this was a logical extension of Experiment 2.

The aims of Experiment 3 were to:

- Produce a range of neural-network models that use a conventional MSE loss function for training.
- Train each model, tracking estimates of the training data p-value randomness and test data empirical validity throughout the training process.
- Examine whether the estimated training data p-value randomness is useful in the training process if the goal is to produce a conformal predictor that has a high level of empirical validity on test data.

6.6.2 Methodology

Raw training and test data generation
For Experiment 3 two types of data were generated: quadratic data and simulated ambulance data. In both cases 100 complete (training and test) datasets were generated. These sets were generated using the same generation rule but with the seed of python's pseudo-random number generator set to each of $0, 1, ..., 99$ beforehand. Using each seed, 100 training examples were generated using indices $i = 1, ..., 100$ and 100 test examples were generated using index $i = 101$. A dataset of size 100 was large enough to give fine-grained confidence results, but small enough to perform the calculations for this experiment in a reasonable time.

Quadratic data generation
For the quadratic data the generation rule was:

- $x_i = i$
- $y_i = x_i^2 + \epsilon_i$, where each ϵ_i was an independent sample from a normal distribution with mean 0 and standard deviation $\frac{i^2}{10}$. Note that this is different to the standard deviation of 1 that used in the quadratic data in Experiment 2. On some initial runs using data generated with a standard deviation of 1 the empirical validity was highly sensitive to model fit, and this was found to be so pronounced with neural networks that it gave extremely volatile results during the model training process. Increasing the standard deviation to $\frac{i^2}{10}$ gave more stable results.

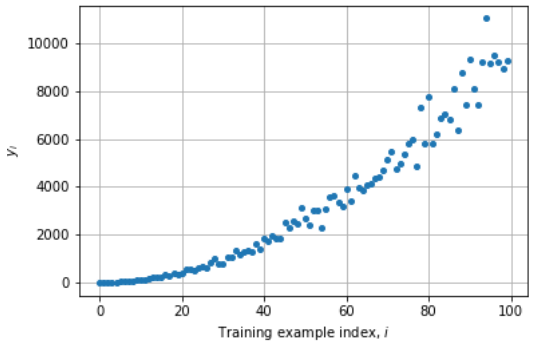
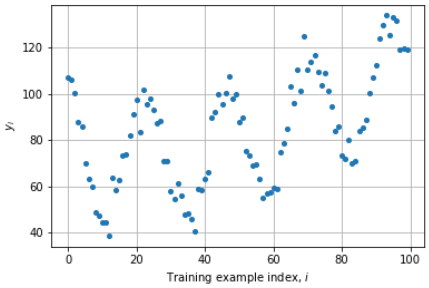


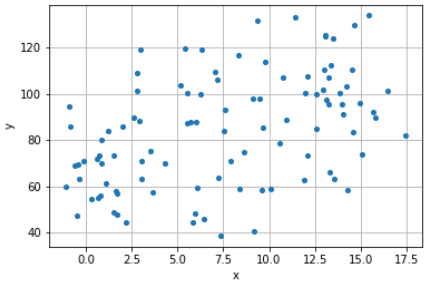
Figure 13: Experiment 3 quadratic training data, with increasing additive Gaussian noise.

Simulated ambulance data generation

The simulated ambulance data was the same as used in Experiment 2 (see Section 6.5.2) and was designed to simulate real-world hourly ambulance demand in a town or city. Details of the generation rule used for the simulated ambulance data are in Appendix B. The simulated ambulance dataset generated with seed 0 is shown in Figure 14.



(a) Simulated ambulance demand (y_i) against time (i), over 100 hours.



(b) Simulated ambulance demand (y) against outdoor temperature (x), over 100 hours.

Figure 14: Experiment 3 simulated ambulance data.

Modelling approach

For each simulated ambulance dataset, window transforms of length 2, 3 and 4 were performed. The windowed data was whitenend using principal component analysis (PCA) as a preprocessing step to simplify the computation and statistical analysis, as explained in [48].

For each of the quadratic datasets, window transforms of length 2 were performed. Transforms of length 3 and 4 were not performed on this data. Representing each windowed training example as a row vector, the columns created by the original raw x_i values are linearly dependent, giving numerically unstable results in the whitening process. Dimensionality reduction is the obvious solution, but is not performed in this work to allow comparisons to be made between the

results from the quadratic data and results from the simulated ambulance data. Additionally, a window size of 2 is sufficient to capture a quadratic trend, so is sufficient for this experiment without further processing.

Neural Network – Feedforward architecture
Fully-connected feedforward neural networks with 0, 1 and 2 hidden layers were used. Each network had an additional affine output layer, so a network with 0 hidden layers corresponded to linear regression, including an offset term. In all cases the non-linear layers consisted of an affine transform followed by a Rectified Linear Unit (ReLU) activation function. For simplicity, each hidden layer had the same number of outputs as inputs. For an input x_i, the output of the final affine layer of each network was scalar prediction $\hat{y}_i$.

The general architecture for the neural networks used in this report is shown in Figure 15.

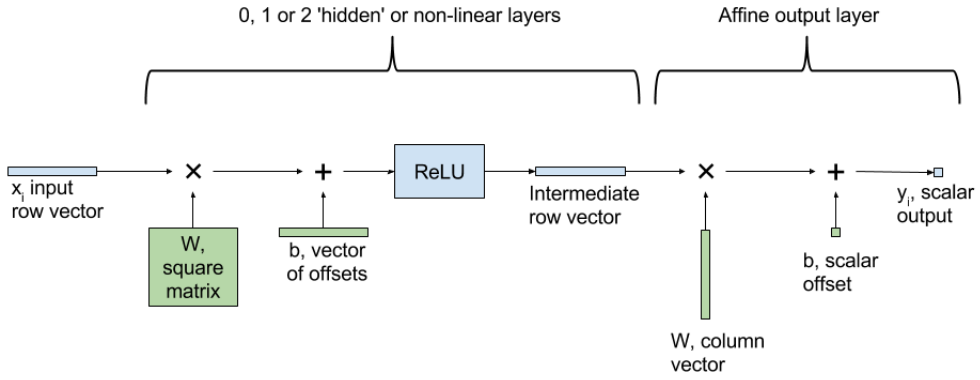


Figure 15: Feedforward neural network architecture used in this report. The model parameters are shown in green.

Neural Network – Initialisation
The W matrices in the neural network were initialised using a He initialisation strategy, which is suited to using ReLU activation functions, as described in [49]. The offsets were all initialised to a value of 0.01, to bias the ReLUs to be active at the start of training.

Neural Network – Training Process
The loss function used in the training process was the mean squared error between the predicted labels $\hat{y}$ and the real labels y. Tensorflow's RMSProp optimiser was used with the default configuration because it is a robust and general-purpose optimiser that performs momentum-based gradient descent. In each case 2000 training steps were performed using a learning rate of 0.001. The entire dataset was small enough to be used as a single batch for training. This configuration was found to train almost all the models to the point where the training error flattened off. While not being best suited to each individual model, using the same hyperparameters across all the models removes a possible source of differences.

A summary of which models were fitted to which dataset types is shown in Table 5.

Test No.	Dataset type	Window Length	Non-lin. layers	Training steps	Learning rate
1	Ambulance	2	0	2000	0.001
2	Ambulance	2	1	2000	0.001
3	Ambulance	2	2	2000	0.001
4	Ambulance	3	0	2000	0.001
5	Ambulance	3	1	2000	0.001
6	Ambulance	3	2	2000	0.001
7	Ambulance	4	0	2000	0.001
8	Ambulance	4	1	2000	0.001
9	Ambulance	4	2	2000	0.001
10	Quadratic	2	0	2000	0.001
11	Quadratic	2	1	2000	0.001
12	Quadratic	2	2	2000	0.001

Table 5: Experiment 3 neural network model fitting configurations.

Neural Network – Calculation of randomness loss during training
During the neural network training process of Experiment 3 the *randomness loss* was recorded, but not used. As with $p_{dev,3}$, randomness loss is an estimate of how far the training data p-values are from being random. Its calculation is described in this section.

In Experiment 2 training data p-values were calculated and used to produce an estimate of (non-)randomness, $p_{dev,3}$. Experiment 3 used the same method, but used *synthetic* p-values in place of the true p-values. The result is called the *randomness loss*, reflecting its use as a loss function for training neural networks.

The use of synthetic p-values was not strictly necessary for Experiment 3, but was required for Experiment 4's neural network training process, so was added to Experiment 3 to allow comparisons to be made.

Synthetic p-values
The calculation of synthetic p-values was as follows, also shown in the workflow diagram in Figure 16. This calculation is performed at every training step.

1. Calculate the training nonconformity scores $\{\alpha_1, ..., \alpha_n\}$.

2. Calculate the p-values $\{p_1, ..., p_n\}$ from the nonconformity scores using a conformal transducer, as described in Section 5.6.

3. Fit a curve relating nonconformity scores to p-values using a quadratic regression model of the form $p_i \approx \beta_0 + \beta_1\alpha_i + \beta_2\alpha_i^2$, minimising the mean-squared error. The regression is constrained to make the offset equal to 1, because the p-value for a nonconformity score of zero is always close to 1. This constraint was implemented by subtracting 1 from all the p-values, fitting a quadratic regression model with no β_0 offset term, and then setting $\beta_0 = 1$. This constraint was found to stabilise the p-value fit.

4. Use the nonconformity scores as inputs to the model to calculate the synthetic p-values.

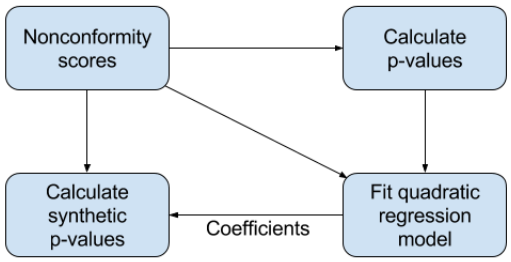
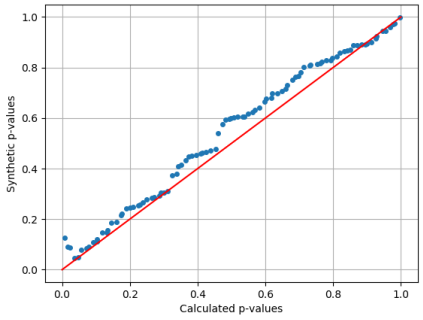


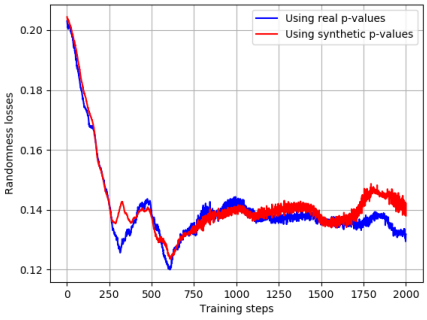
Figure 16: Workflow for calculation of synthetic p-values.

While the synthetic p-values were different to the real p-values, the method of constrained quadratic regression was found to give synthetic p-values consistently close to the real p-values. This was desirable, because it meant that the randomness loss used on Experiments 3 and 4 would behave similarly to how $p_{dev,3}$ would have behaved.

An illustration of this similarity on synthetic ambulance data is shown in Figure 17 below, where a window length of 4 and a neural network with one hidden layer was used. Figure 17(a) shows that, at the end of training, the synthetic p-values were broadly similar to the actual p-values. More importantly, Figure 17(b) shows how the randomness loss (as defined, using synthetic p-values), in red, evolved during training. It can be seen to behave similarly to how the randomness loss would behave if calculated using actual p-values, shown in blue.



(a) Comparison of real and synthetic p-values, after 2000 training steps



(b) Impact of using synthetic p-values on randomness loss during the training process

Figure 17: Comparison between use of real and synthetic p-values for training a neural network model with one hidden layer on simulated ambulance data.

Neural network – Computation graph

Neural networks implemented in the Tensorflow python package are constructed through the use of a *computation graph*. Calculations within the computation

graph are efficiently implemented, and the neural network training process takes place within the computation graph. The calculation of the randomness loss using synthetic p-values was incorporated into the neural network's computation graph as shown in Figure 18 below.

The computation graph was the reason behind the creation of the synthetic p-values. Calculations in a computation graph must be smooth (i.e. differentiable) for the backpropagation training process to work. The training data p-value calculation as presented in (4) was not differentiable as it did not vary smoothly with changing nonconformity scores. To solve this issue, the randomness loss using the synthetic p-values was created, which is differentiable.

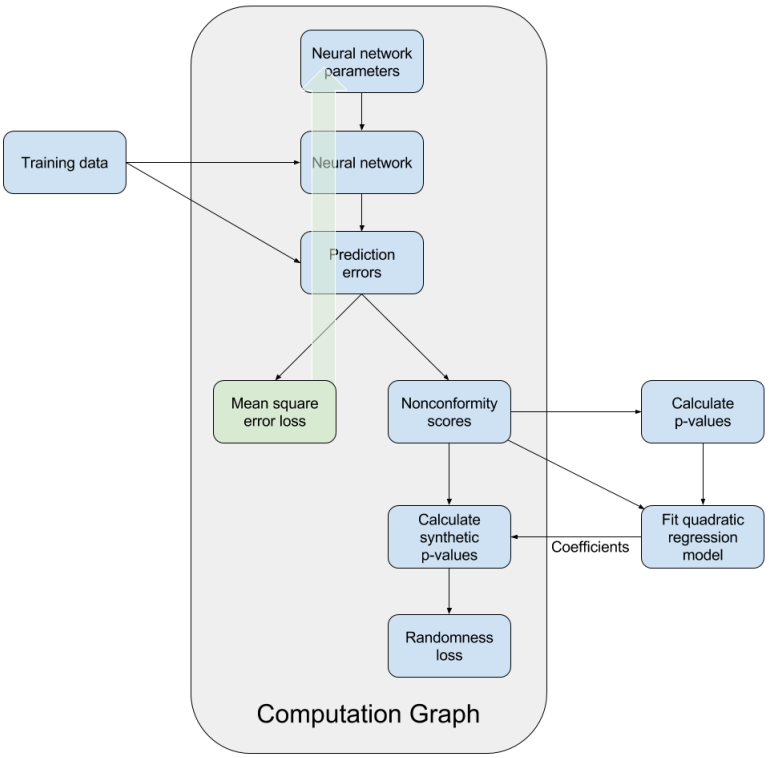
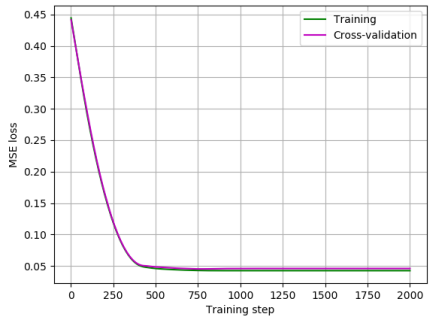


Figure 18: Experiment 3 neural network training architecture. The gradient (shown as a thick green arrow) flows back from the mean squared error loss to the neural network's parameters during the backpropagation training process. The randomness loss using synthetic p-values is calculated in the computation graph but not used for training.

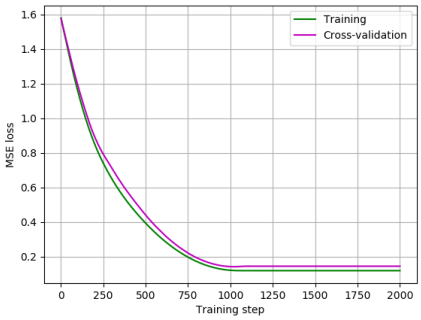
6.6.3 Results

The randomness loss and validity gap are presented below in Figures 19, 20 and 21 for the quadratic and simulated ambulance datasets generated with seed 0. Only the simulated ambulance results using a window length of 4 are presented, as they were similar to the results with window lengths of 2 and 3. The training and 5-fold cross validation MSE loss during the training process are also presented, to give visibility of which models are tending to overfit. The same results for datasets generated using seeds 1 and 2 are presented in Appendix A.2 for comparison.

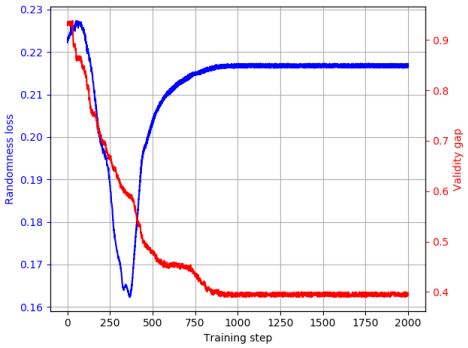
Neural networks with 0 hidden layers



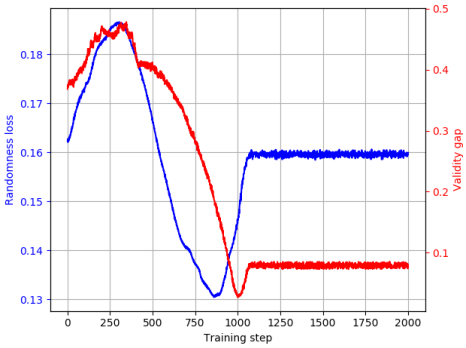
(a) Quadratic data.



(b) Simulated ambulance data.



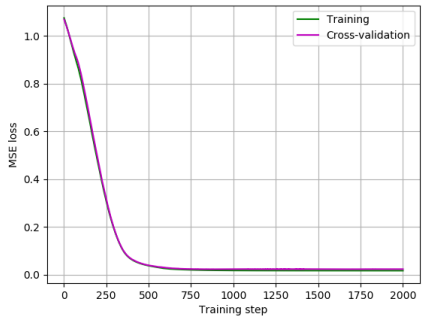
(c) Quadratic data.



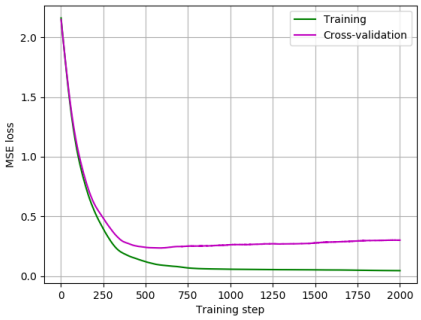
(d) Simulated ambulance data.

Figure 19: Evolution of training and 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 0 hidden layers. The quadratic data was generated with random seed 0 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 0 and had a window transform of length 4.

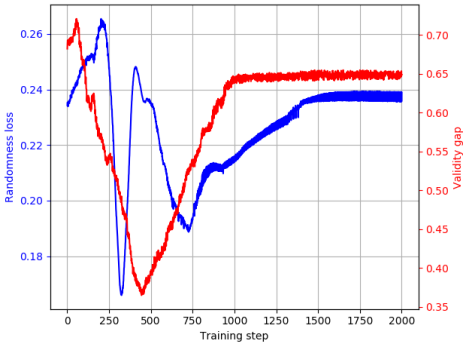
Neural networks with 1 hidden layer



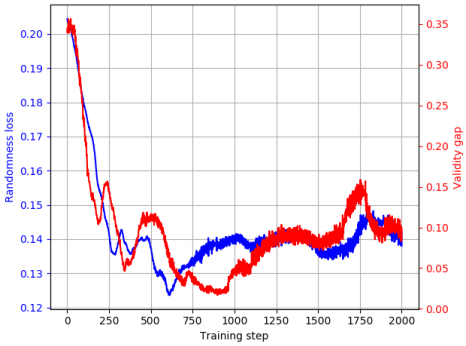
(a) Quadratic data.



(b) Simulated ambulance data.



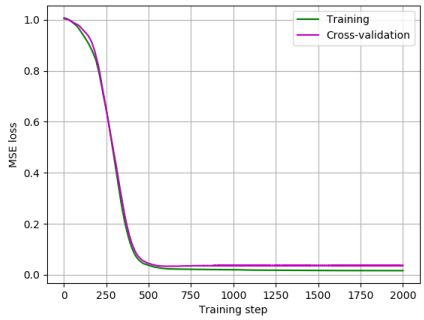
(c) Quadratic data.



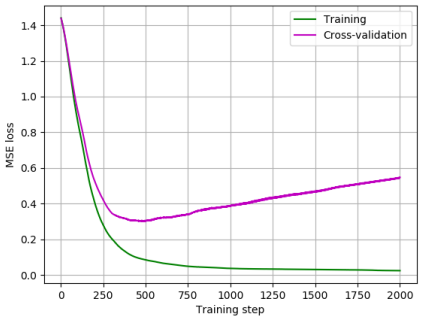
(d) Simulated ambulance data.

Figure 20: Evolution of training and 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 1 hidden layer. The quadratic data was generated with random seed 0 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 0 and had a window transform of length 4.

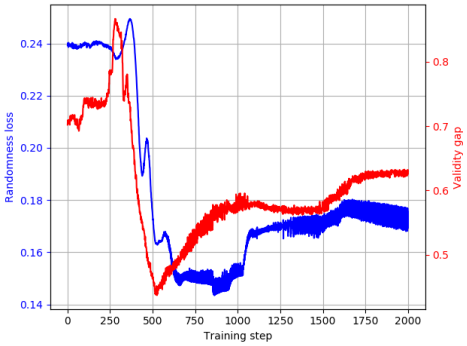
Neural networks with 2 hidden layers



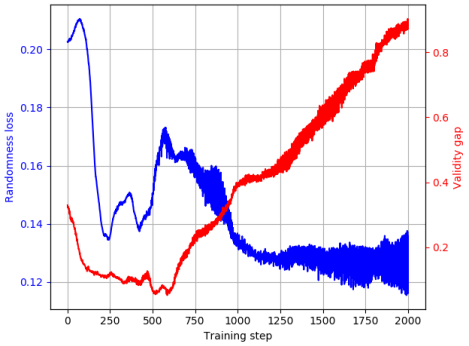
(a) Quadratic data.



(b) Simulated ambulance data.



(c) Quadratic data.



(d) Simulated ambulance data.

Figure 21: Evolution of training and 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 2 hidden layers. The quadratic data was generated with random seed 0 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 0 and had a window transform of length 4.

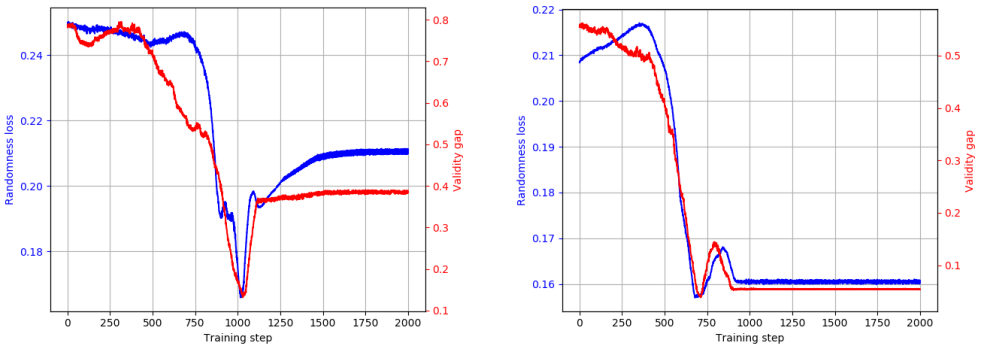
6.6.4 Analysis

Neural networks with 0 hidden layers

Examining the plots shown both in Figure 19 and in Appendix A.2, on both the quadratic and the simulated ambulance datasets the cross-validation loss flattened during the training process and did not rise again with further training. The only exception to this was when the MSE loss was still decreasing, indicating that the training process required more than 2000 steps. At around the same point in training as the cross-validation loss flattened, the randomness loss and validity gap also flattened.

On the plots shown the minimum validity gaps for the quadratic data were in the range 0.02 to 0.40, and were on average much higher than for the simulated ambulance data, which had minimum validity gaps below 0.08. Across both datasets, in four out of the six cases shown the validity gap reached its minimum close to or shortly after the randomness loss reached its minimum. Examining the results across the 100 datasets generated with seeds $0, ..., 99$, this was found

to occur in 73 cases out of 100 on the quadratic data and 74 cases out of 100 on the simulated ambulance data. There were many striking examples of similarity between the randomness loss and validity gap such as those shown in Figure 22:



(a) Quadratic data, seed = 64 (b) Simulated ambulance data, seed = 32

Figure 22: Examples of particularly high levels of similarity between the randomness loss and the validity gap for both quadratic data and simulated ambulance on neural networks with 0 hidden layers.

Neural networks with 1 hidden layer

Examining the plots shown in Figure 20 and in Appendix A.2, the cross-validation loss flattened on the quadratic data but tended to rise as training continued on the simulated ambulance data. There did not seem to be any pattern in the flatness of the randomness loss and the validity gap.

On the plots shown the validity gaps for the quadratic data were, in general, much higher and more varied than for the simulated ambulance data. The minimum validity gaps achieved during the training on the quadratic data were in the range 0.05 to 0.37, whereas for the simulated ambulance data the range was around 0.02 to 0.03. As with the networks with 0 hidden layers, across both datasets in four out of the six cases shown the validity gap reached its minimum shortly after the randomness loss reached its minimum, but on checking on 100 datasets generated with seeds $0, ..., 99$ this was not found to be the case in general. However, as with the networks with 0 hidden layers, in many cases there was a similarity between the shape of the randomness loss and the validity gap.

Neural networks with 2 hidden layers

Examining the plots in Figure 21 and in Appendix A.2, on the simulated ambulance data the cross-validation loss always reached a minimum and then rose again. On the quadratic data the cross-validation loss flattened out. In all cases shown where the cross-validation reached a sharp minimum, the validity gap reached its minimum at around the same point. After this point there was a rise in validity gap and a marked increase in the volatility of the randomness loss. Examining the 100 datasets generated using seeds $0, ..., 99$ this pattern occurred in almost every simulated ambulance dataset. The quadratic data showed more variation, with no clear trend discernible.

6.6.5 Discussion

Neural networks with 0 hidden layers
For the networks with 0 hidden layers the flattening effect seen on the randomness loss and validity gap occur because the parameter updates are small in each training step, so the model itself is only changing slightly. This is consistent with the model being, in essence, linear regression, so the challenge of finding the parameter values that minimise the training MSE loss is known to have a single solution that is slowly changing around the global minimum.

In four out of the six cases shown (and 73 times out of 100 on the full set of results) the validity gap reaches its minimum around or shortly after the the point where the randomness loss reaches its minimum. Happening at the same time would be consistent with the tentative link established between $p_{dev,3}$ and validity gap in Experiment 2. It is not clear why validity gap would reach its minimum shortly after the minimum in randomness loss, rather than at the same point during training. However, this is an interesting phenomenon and potentially can be used to decide when to stop training models that have a tendency to underfit to give best empirical validity. This could be a useful tool, because these models underfit in the conventional sense so cross-validation does not give insight about when to stop training. As these models cannot overfit, this will not prevent overfitting, but could save computational resources by helping indicate when to stop training. Flattening of the training MSE error could also be used for this, but the flattening of randomness loss also appears to be occurring more sharply, making it a clearer indicator.

Neural networks with 1 hidden layer
For the networks with 1 hidden layer there do not seem to be strong patterns emerging. It would appear that, for the particular choice of datasets and model architectures, some datasets are able to be well modelled and some are not. The similarity that is apparent in many cases between the shape of the randomness loss curve and the validity gap curve gives a further hint that training data p-value randomness and test data empirical validity may be connected, at least on the two types of datasets considered in this work.

Neural networks with 2 hidden layers
For the networks with 2 hidden layers there seems to be a clear relationship between overfitting and volatility of randomness loss. This makes sense because, as the prediction errors become small the ranking order of nonconformity scores (and so the p-values and thus the randomness loss) becomes very sensitive to changes in the model parameters. A potentially more useful result, however, is the fact that the lowest validity gaps are often achieved at the same point during the training process as the lowest MSE cross-validation error. This provides a possible additional use for this already widely-used technique.

6.6.6 Conclusion

On the two datasets used, randomness loss was found to be a useful quantity to monitor when choosing when to stop training models that have a tendency to underfit the data, and so for which minimising cross-validation loss is not an

option. For these models the lowest validity gap tended to occur at around the same point in the training process as a flattening out of randomness loss. For models that were capable of overfitting, a highly volatile randomness loss was seen to indicate overfitting, but not to be any more informative than the rising MSE cross-validation loss. The usual method of selecting the number of training steps by minimising cross-validation loss was seen also to give close to the minimum validity gap.

6.7 Experiment 4

6.7.1 Introduction

Experiment 4 was an investigation into whether randomness loss is useful for the training of neural networks. The aims of Experiment 4 were to:

1. Reuse the neural-network models from Experiment 3, incorporating training data p-value randomness (as approximated by the randomness loss) into their loss functions.

2. Train the models, comparing each model's validity gap with the validity gap achieved by the equivalent model in Experiment 3.

Experiment 4 used exactly the same datasets and window transforms as Experiment 3. This section contains a summary of this information but more details can be found in Experiment 3, Section 6.6.2. Experiment 4 only differed from Experiment 3 in the loss function used to train the neural networks.

6.7.2 Methodology

Raw training and test data generation
For Experiment 4 the same two types of data were generated as for Experiment 3: quadratic data and simulated ambulance data. In both cases 100 complete (training and test) datasets were generated. These sets were generated using the same generation rule but with the seed of python's pseudo-random number generator set to each of $0, 1, ..., 99$ beforehand. Using each seed, 100 training examples were generated using indices $i = 1, ..., 100$ and 100 test examples were generated all using index $i = 101$.

Quadratic data generation
For the quadratic data the generation rule was:

- $x_i = i$
- $y_i = x_i^2 + \epsilon_i$, where each ϵ_i was an independent sample from a normal distribution with mean 0 and standard deviation $\frac{i^2}{10}$.

Simulated ambulance data generation
The simulated ambulance data was the same as that used in Experiments 2 and 3, and was designed to simulate real-world hourly ambulance demand in a town or city. Full details are given in Appendix B

Modelling approach
For each simulated ambulance dataset, window transforms of length 2, 3 and 4 were performed. The windowed data was PCA-whitenend as a preprocessing step. For each of the quadratic datasets, a window transform of length 2 was used.

Neural Network – Feedforward architecture
Fully-connected feedforward neural networks with 0, 1 and 2 hidden layers were

used. Each network had an additional affine output layer, so a network with 0 hidden layers corresponded to linear regression, including an offset. In all cases the non-linear layers consisted of an affine transform followed by a ReLU activation function.

Neural Network – Initialisation
In all cases the matrices in the neural network were initialised using a He initialisation strategy, which is suited to using ReLU activation functions, as described in [49]. The offsets were all initialised to a value of 0.01, to bias the ReLUs to be active at the start of training.

Neural Network – Training Process
Experiment 4 used a loss function which combines the MSE loss, as used in Experiment 3, with the randomness loss, as described in 6.6.2. Based on the results of some initial trials, the losses were weighted as

$$Loss_{total} = Loss_{MSE} + 0.1 \times Loss_{Randomness}. \tag{11}$$

A coefficient of 0.0 multiplying the randomness loss in (11) would recover Experiment 3. As with Experiment 3, Tensorflow's RMSProp optimiser was used for 2000 training steps with a learning rate of 0.001.

A summary of which models were fitted to which dataset types is shown in Table 6. This is the same set of models as Experiment 3.

Test No.	Dataset type	Window Length	Non-lin. layers	Training steps	Learning rate
1	Ambulance	2	0	2000	0.001
2	Ambulance	2	1	2000	0.001
3	Ambulance	2	2	2000	0.001
4	Ambulance	3	0	2000	0.001
5	Ambulance	3	1	2000	0.001
6	Ambulance	3	2	2000	0.001
7	Ambulance	4	0	2000	0.001
8	Ambulance	4	1	2000	0.001
9	Ambulance	4	2	2000	0.001
10	Quadratic	2	0	2000	0.001
11	Quadratic	2	1	2000	0.001
12	Quadratic	2	2	2000	0.001

Table 6: Experiment 4 neural network model fitting configurations.

Neural network – Computation graph
The neural networks used in Experiment 4 calculated MSE loss, randomness loss and total loss inside their computation graphs. During the training process both loss types contributed to the gradient flow back to the model parameters as part of the backpropagation training process, as shown in Figure 23, below.

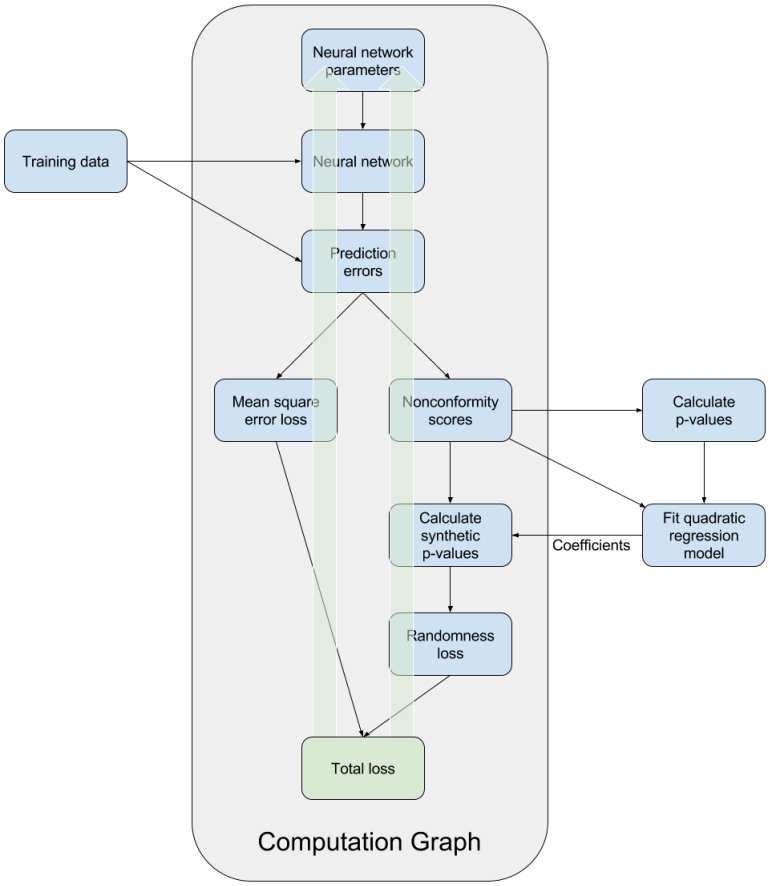
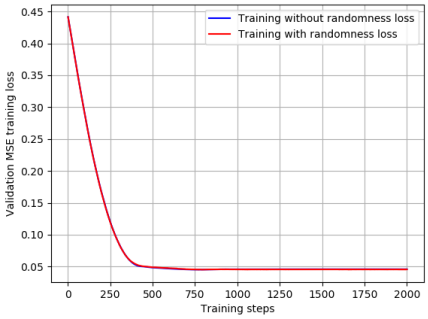


Figure 23: Experiment 4 neural network training architecture. The gradient (shown as thick green arrows) flows back via two separate paths from the total loss to the neural network's parameters during the backpropagation training process.

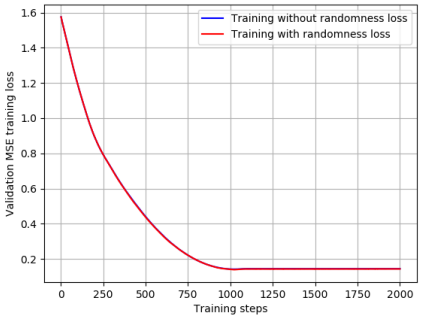
6.7.3 Results

The MSE validation loss, randomness loss and validity gap for training both with and without the randomness loss are shown below.

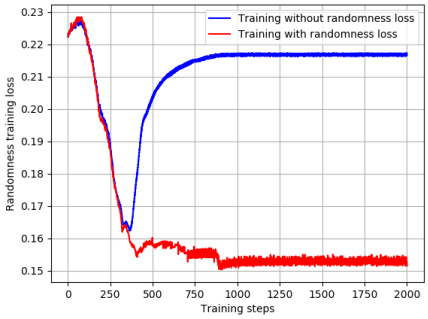
Neural networks with 0 hidden layers



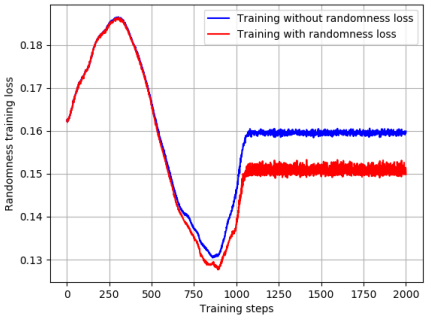
(a) Quadratic data.



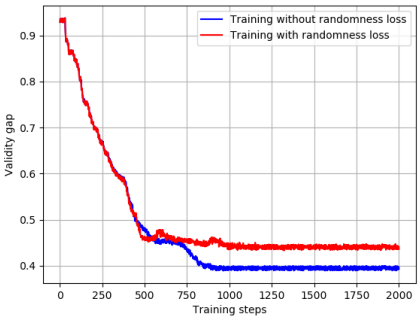
(b) Simulated ambulance data.



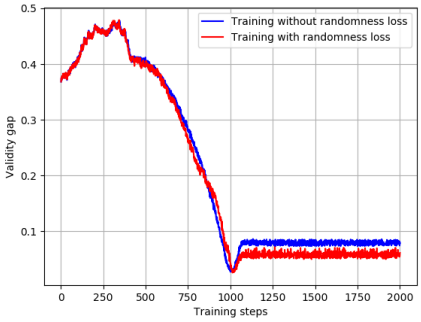
(c) Quadratic data.



(d) Simulated ambulance data.



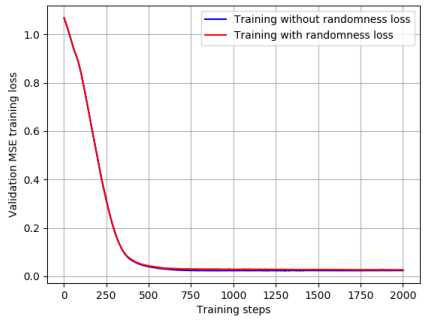
(e) Quadratic data.



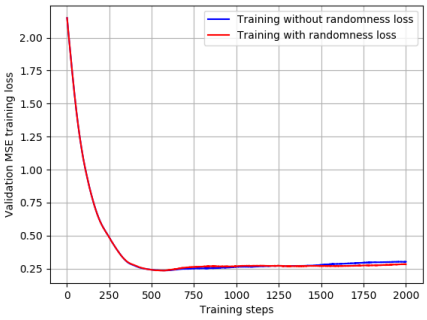
(f) Simulated ambulance data.

Figure 24: Comparison of the evolution of 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 0 hidden layers, with and without the use of randomness loss in the neural network's loss function. The quadratic data was generated with random seed 0 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 0 and had a window transform of length 4.

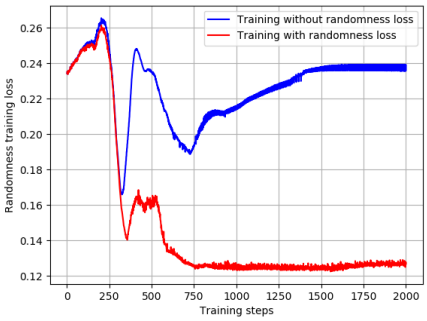
Neural networks with 1 hidden layer



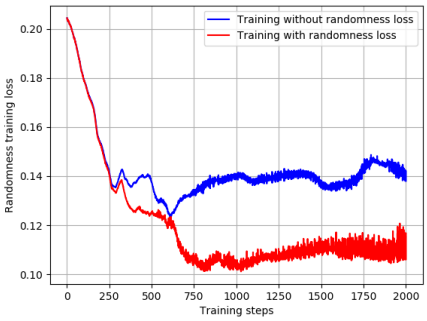
(a) Quadratic data.



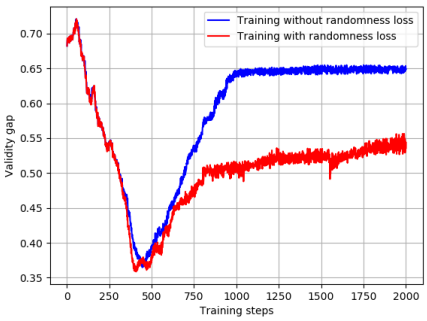
(b) Simulated ambulance data.



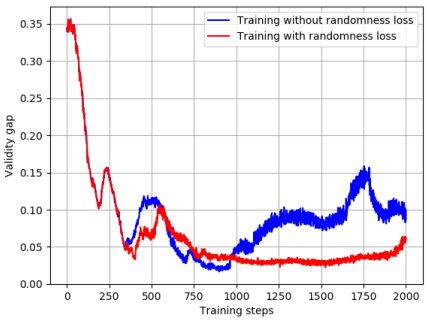
(c) Quadratic data.



(d) Simulated ambulance data.



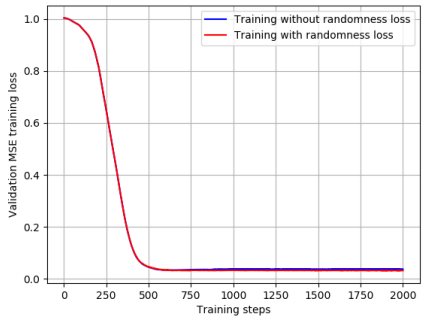
(e) Quadratic data.



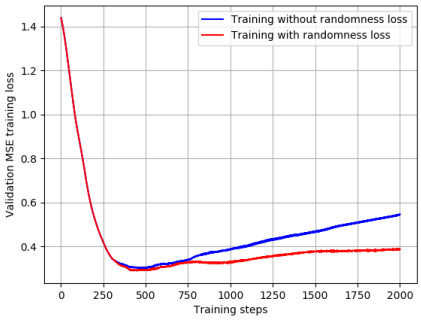
(f) Simulated ambulance data.

Figure 25: Comparison of the evolution of 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 1 hidden layer, with and without the use of randomness loss in the neural network's loss function. The quadratic data was generated with random seed 0 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 0 and had a window transform of length 4.

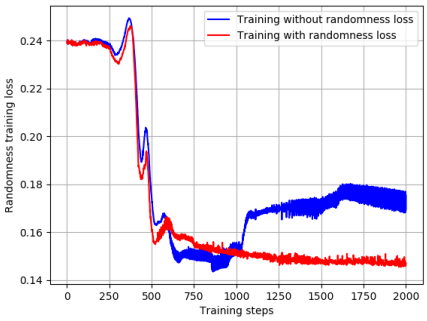
Neural networks with 2 hidden layers



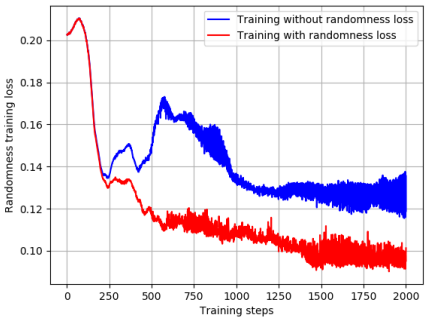
(a) Quadratic data.



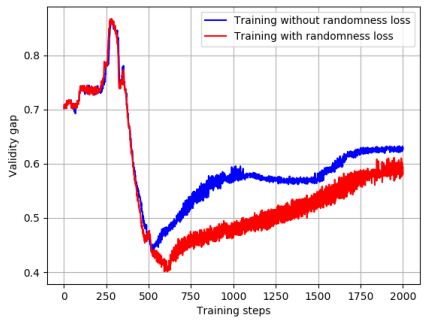
(b) Simulated ambulance data.



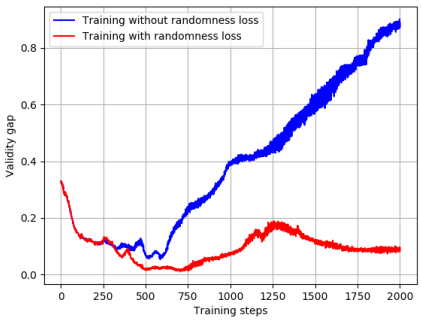
(c) Quadratic data.



(d) Simulated ambulance data.



(e) Quadratic data.



(f) Simulated ambulance data.

Figure 26: Comparison of the evolution of 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 2 hidden layers, with and without the use of randomness loss in the neural network's loss function. The quadratic data was generated with random seed 0 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 0 and had a window transform of length 4.

Test No.	Validity gap change		% Validity gap change	
	Mean	Spread	Mean	Spread
1	-0.00047	± 0.01493	-0.35	± 11.34
2	-0.00719	± 0.03547	-5.30	± 26.14
3	-0.00370	± 0.06362	-2.44	± 42.02
4	0.00220	± 0.01792	2.27	± 18.53
5	-0.00399	± 0.11388	-2.28	± 65.22
6	-0.04313	± 0.14780	-15.97	± 54.73
7	-0.00517	± 0.01934	-5.87	± 21.95
8	-0.05738	± 0.14000	-17.39	± 42.43
9	-0.14573	± 0.1823 0	-28.83	± 36.07
10	-0.02759	± 0.04534	-7.90	± 12.98
11	-0.02949	± 0.07370	-7.59	± 18.98
12	-0.03289	± 0.10211	-7.99	± 24.81

Table 7: Average change over 100 randomly generated datasets of validity gap (see Section 6.3.5) after 2000 training steps, due to incorporating the randomness loss into each model's neural network loss function. The model details and training configuration of each test are detailed in Section 6.7.2. The spread is the root mean square (RMS) deviation from the mean.

6.7.4 Analysis

Compared to Experiment 3, Experiment 4 showed a reduction in randomness loss in all models across both dataset types.

The minimum validity gap achieved during the training of each model in Experiment 4 did not seem to be consistently lower than the equivalent in Experiment 3, but across the 100 generated datasets was rarely higher. The more complex models that exhibited overfitting behaviour showed a lower increase in validity gap during the overfitting region of training on Experiment 4 than on Experiment 3. A clear example of this was the neural network with 2 hidden layers trained on simulated ambulance data with window length 4, as shown in Figure 26 (f). The inclusion of the randomness loss in the training process reduced the validity gap after 2000 training steps from around 0.9 (Experiment 3 result) to around 0.1 (Experiment 4 result). This was an unusually large improvement, but even across the 100 generated datasets the average improvement was 28.83%, as shown in the Test 9 entry in Table 7. Figure 26 (b) shows that the inclusion of randomness loss also in this case reduced the amount of overfitting of the model, which was a common observation across the 100 generated datasets.

6.7.5 Discussion

The reduction in randomness loss compared with Experiment 3 is entirely expected, because the neural network is now incentivised to reduce it. However, it is encouraging to see that it is indeed happening, indicating that the neural network training process is functioning correctly.

On the two datasets used in this experiment, the inclusion of randomness loss in the neural network loss function does not seem to bring any consistent benefit to the models that have a tendency to underfit. The validity gap that is achieved is only improved on average by a few percent (as shown in Table 7), but with a spread that is much larger than this improvement, meaning any performance benefit would be unreliable. It is not surprising that the randomness loss does not bring any benefit in this case: these simple models are not even able to fully capture the main trend of the data, so the nonconformity measures are likely to have a large range.

At the other end of the spectrum, including randomness loss in the neural network loss function does seem to have a role to play in the training of more flexible models. It seems to be helping to prevent the models from overfitting. It also seems to be preventing or suppressing the rise in validity gap in the overfitting period of training, acting as a kind of 'empirical regulariser'. The addition of the randomness loss seems not to affect the initial model fitting, perhaps because the MSE loss dominates the randomness loss early on. When the minimum validity gap is found the MSE loss is close to zero and the randomness loss seems to encourage the model to keep the validity gap low on further training. One possible explanation for this is that, as also observed on Experiment 2, once a model is fitting well the randomness loss seems to behave very differently to MSE training loss, so when the two are combined it brings some extra diversity to the model parameter updates, thus acting as a form of regularisation. The two losses seem to work well together. At the start the MSE loss dominates and finds an approximate

fit, and later the randomness loss holds the fit at a point where it has reasonable empirical validity.

6.7.6 Conclusion

For the two dataset types studied a weighting of 0.1 randomness loss was added to the conventional MSE loss to create a combined loss function for the fitting of neural networks models, which were used to form conformal transducers. This modification was found not to affect the performance of simpler models (neural networks with 0 or 1 hidden layers) in any obviously useful way. On more flexible models (neural networks with 2 hidden layers) the minimum validity gap tended not to be affected. In the training region after the minimum validity was achieved the inclusion of randomness loss had the effect of reducing the tendency of the models to overfit, and suppressed the increase of the validity gap.

7 Conclusion

Experiment 1 confirmed that data generated to be iid gave a level of training data p-value randomness that was consistent with random shuffling, and a validity gap that was very small. Both results were consistent with the theory of conformal prediction.

The results of **Experiment 2** suggested a link between training data p-value randomness and test data empirical validity on data that violated the exchangeability assumption. During the early stages of model training, large reductions were seen on all the $p_{dev,x}$ estimates of training data p-value (non-)randomness and the validity gap. Once the models started to fit well, the pattern of improvement broke down, and in some cases the model worsened on both measures. On the validity gap there seemed to be a phenomenon similar to overfitting at play, despite the models used being unable to overfit in the conventional sense.

Building on these observations, **Experiment 3** found that training data p-value randomness could be used as part of an early-stopping strategy for the training of neural networks to target the smallest validity gap. For simple neural networks this strategy would be to stop training when the randomness loss flattens off. For more complex neural networks the strategy would be to stop training before the randomness loss becomes very volatile. For the second case, however, there did not seem to be an advantage over the usual method of finding the minimum cross-validation MSE loss.

Experiment 4 integrated the training data p-value randomness into the neural network training process. This did not prove to have a useful effect on the simpler neural network models. On the more complex models the validity gap was not reduced during the early stages of training, and nor was the minimum validity gap reduced. However, there was a large suppression of the subsequent increase of validity gap during the period of training where the models were overfitting, demonstrating an increased level of robustness of the models' validity gaps to overtraining.

7.0.1 Review of results

In summary, the principal results of this work were:

1. There were signs of a connection between training data p-value randomness and test data empirical validity, but the precise nature of this connection remains unclear.

2. Simple neural network models that underfit (in terms of MSE loss) were found to have a $p_{dev,3}$ estimate and validity gap that showed a marked plateau and stopped changing with further training. The flattening of $p_{dev,3}$ could be used as a signal of when to stop training, and would be a sharper indication for this than the levelling off of the training MSE error.

3. When training more complex neural network models that tended to overfit (in terms of MSE loss), during the overfitting phase the training data p-value randomness became extremely volatile. This did not seem to offer any additional insight beyond conventional cross-validation on when to stop training.

4. Incorporating a small amount of randomness loss into the neural network's loss function gave, on average, a large increase in the model's robustness to overfitting in terms of empirical validity. The tendency to overfit in terms of MSE loss was also reduced.

7.0.2 Review of research question

The research question for this work, posed in Section 2.5 was:

> *Given a dataset that is not probabilistically exchangeable, are there techniques to train neural networks in order to form conformal predictors that are likely to produce empirically valid confidence predictions on future test data?*

This work found that incorporating the randomness loss into the neural network training process did not, in general, improve the 'best' model that the neural network could produce, in terms of either empirical validity or prediction error. However, incorporating the randomness loss made the models remain close to their best empirical validity over a much larger range of training steps, considerably simplifying the training task if empirical validity is required.

Using this method on the datasets studied would make a given level of empirical validity easier to achieve, and so more likely, which answers the research question.

8 Suggestions for further work

This work produced promising results but was an empirical study on a small number of datasets. A focus of further work could be to perform a much broader and more detailed empirical study. Once a clearer empirical picture is established, this would direct the focus of theoretical work with the aim of putting the ideas proposed in this work on a principled mathematical footing.

Specific areas for a further empirical study would be:

- Investigate the relationship between training data p-value randomness and test data empirical validity on a much wider variety of datasets, including datasets generated in an adversarial manner.
- Investigate different methods of measuring training data p-value randomness.
- Investigate methods of approximating the training data p-value randomness in a smooth manner that improve upon the constrained quadratic regression used in this work.
- Combine the randomness loss with other common regularisation methods used when training neural networks to see whether they interact in a complementary fashion.

9 Abbreviations and Glossary

9.1 Abbreviations

IID or iid	Independent and identically distributed.
MSE	Mean square error.
NFL	No free lunch.
PCA	Principal component analysis.
ReLU	Rectified linear unit.
RMS	Root mean square.
SLT	Statistical learning theory.

9.2 Glossary

Computation graph	The construct in *Tensorflow* by which computations are represented.
Conformal transducer	Converts a set of data into a set of *p-values* by comparing *nonconformity scores*.
Conformal predictor	A method of prediction that can deliver predictive confidence intervals together with predictions.
Empirical risk	A measure of how well a model fits a specific training dataset.
Example	An $object, label$ pair that comprises one datapoint.
Exchangeable	A sequence is exchangeable if every ordering of the sequence is equally likely.
Generation rule	A method used to generated both training data and test data.
Hedged prediction	A prediction that comes with information regarding the confidence the predictor has in the prediction.
Nonconformity score	A measure of how different an *example's* label is to its predicted label.
P-values	A number betwen 0 and 1 used in statistics to indicate the significance of a result. The smaller the number the higher the level of significance.
Point predictions	A prediction that is a single number (in the regression setting) or single class (in the classification setting).
Randomness loss	An approximate measure of the randomness of the order of the *training data p-values*.
Raw test examples	An $object, label$ test *example* where both the object and the label are scalar.
Raw training examples	An $object, label$ training *example* where both the object and the label are scalar.
Risk	A measure of how well a model fits the distribution that generated the training data.

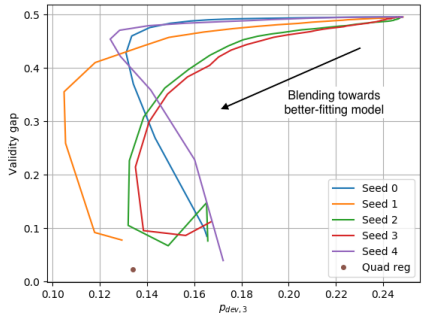
Synthetic p-value	An approximation of a *p-value* based on a *nonconformity score* and a quadratic regression model.
Tensorflow	An open-source machine learning software library.
Test examples	See *windowed test examples*.
Test p-values	*P-values* calculated on the test dataset by a *conformal transducer*.
Training examples	See *windowed training examples*.
Training data p-values	*P-values* calculated on the training dataset by a *conformal transducer*.
Valid	A *conformal predictor* is valid if it produces confidence intervals that contain the true value with the appropriate probability. For example, a 95% confidence interval is valid if it contains the true value 95% of the time. A *conformal transducer* is valid if it produces p-values that are uniformly distributed in the interval [0, 1].
Validity gap	The average deviation of a set of *test p-values* from the *validity line*, indicating how far the test data is from being empirically valid.
Validity line	The $1-\epsilon$ line on a *validity plot* that is traced by the *p-values* if the *p-values* are uniformly distributed in the interval [0, 1].
Validity plot	A plot of a set of *p-values* on the horizontal axis with $1-\epsilon$ on the vertical access, where ϵ is the significance level.
Window transform	The transformation applied to *raw test examples* (resp. *raw training examples*) to convert them into *windowed test examples* (resp. *windowed training examples*).
Windowed test example	An $object, label$ test *example* where the object may be a scalar or may be a vector made from a set of previous *raw examples*.
Windowed training example	An $object, label$ training *example* where the object may be a scalar or may be a vector made from a set of previous *raw examples*.

A Results

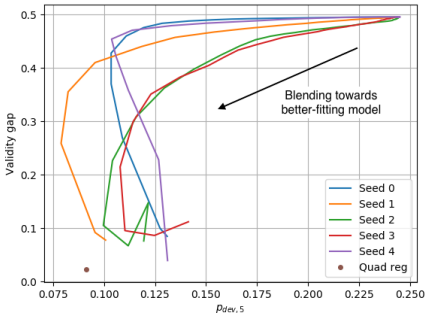
A.1 Experiment 2 Results

A.1.1 Quadratic Data

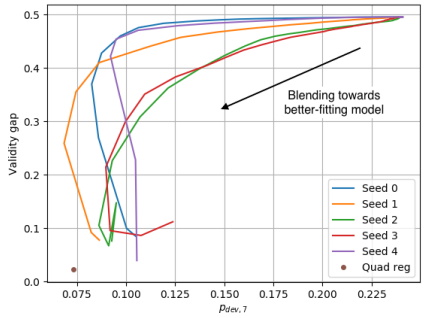
Figure 27 shows the trajectory through the validity gap / $p_{dev,x}$ space of linear model performance on quadratic data as the model's coefficients are blended from a poorly-fitting model to a well-fitting model, for each of $x = \{3, 5, 7, 9\}$. Five different datasets are shown, generated with different random seeds. Validity gap is an estimate of how far the test p-values are from being empirically valid, as detailed in Section 6.3.5. $p_{dev,3}$ is an estimate of how far the training data p-values are from being random, as detailed in Section 6.3.4. The point value of a quadratic regression model is also plotted.



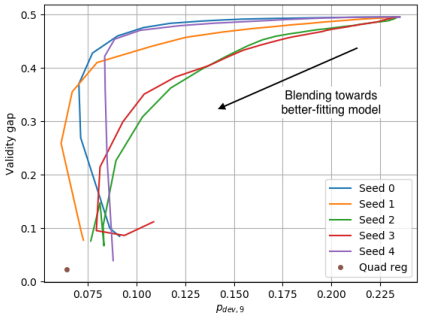
(a) Evolution of the validity gap and $p_{dev,3}$ during the blending process. *Quad reg* is the point result from quadratic regression



(b) Evolution of the validity gap and $p_{dev,5}$ during the blending process. *Quad reg* is the point result from quadratic regression



(c) Evolution of the validity gap and $p_{dev,7}$ during the blending process. *Quad reg* is the point result from quadratic regression

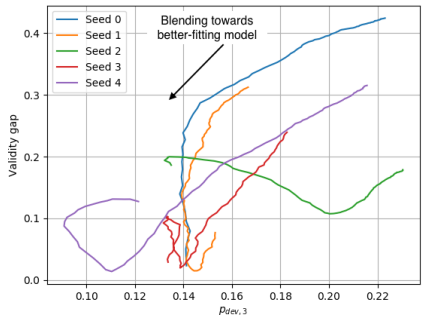


(d) Evolution of the validity gap and $p_{dev,9}$ during the blending process. *Quad reg* is the point result from quadratic regression

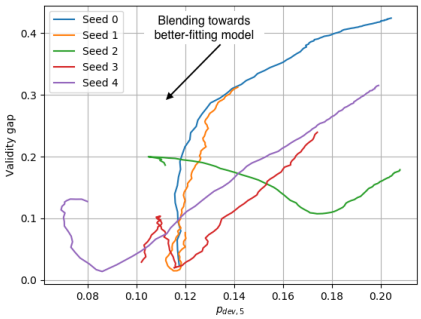
Figure 27: Experiment 2 results

A.1.2 Simulated Ambulance Data

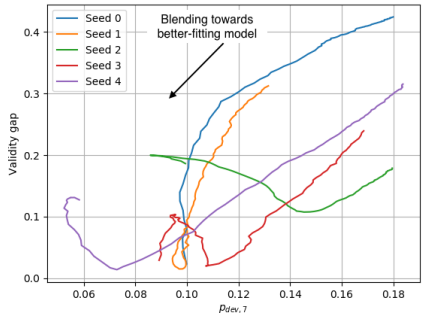
Figure 28 shows the trajectory through the validity gap / $p_{dev,x}$ space of linear model performance on simulated ambulance data as the model's coefficients are blended from a poorly-fitting model to a well-fitting model, for each of $x = \{3, 5, 7, 9\}$. Five different datasets are shown, generated with different random seeds. Validity gap is an estimate of how far the test p-values are from being empirically valid, as detailed in Section 6.3.5. $p_{dev,3}$ is an estimate of how far the training data p-values are from being random, as detailed in Section 6.3.4.



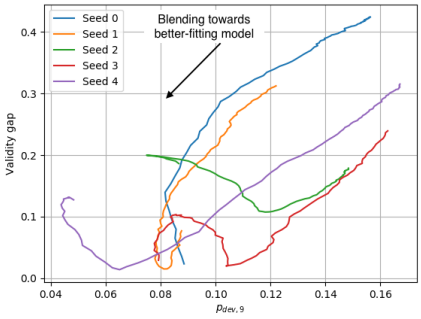
(a) Evolution of the validity gap and $p_{dev,3}$ during the blending process.



(b) Evolution of the validity gap and $p_{dev,5}$ during the blending process.



(c) Evolution of the validity gap and $p_{dev,7}$ during the blending process.



(d) Evolution of the validity gap and $p_{dev,9}$ during the blending process.

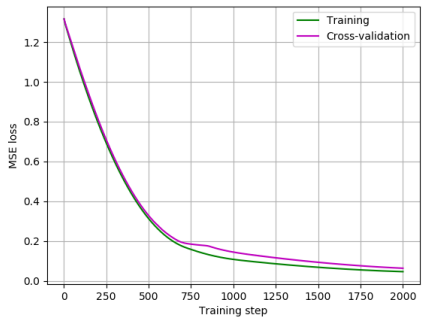
Figure 28: Experiment 2 results

A.2 Experiment 3 Results

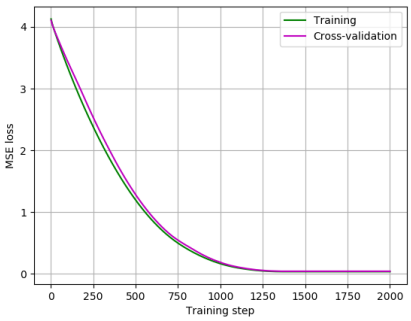
Section 6.6.3 presented results for quadratic and simulated ambulance datasets generated using a random seed of 0. This appendix shows the equivalent results for data generated with seeds of 1 and 2.

A.2.1 Neural networks with 0 hidden layers

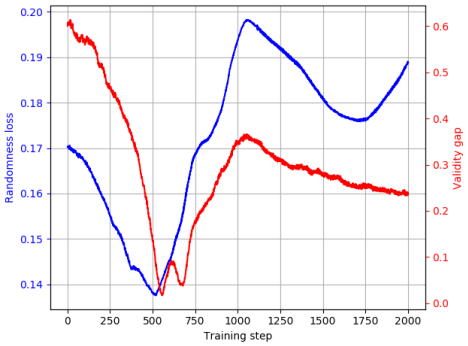
Random seed = 1



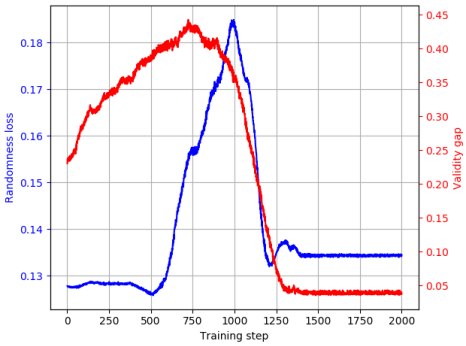
(a) Quadratic data.



(b) Simulated ambulance data.



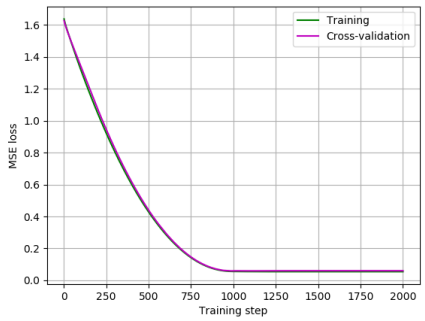
(c) Quadratic data.



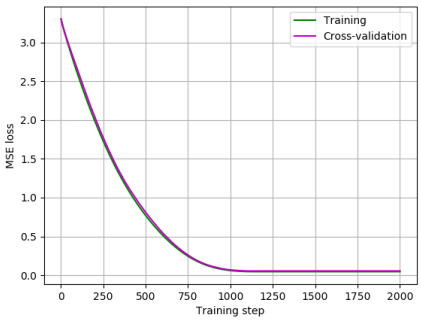
(d) Simulated ambulance data.

Figure 29: Evolution of training and 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 0 hidden layers. The quadratic data was generated with random seed 1 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 1 and had a window transform of length 4.

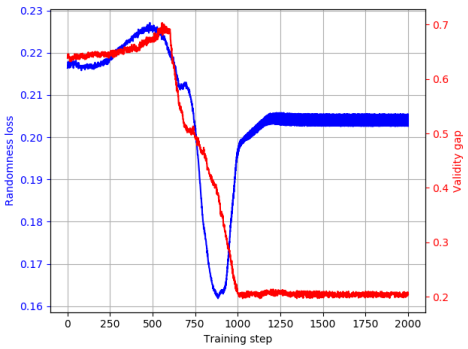
Random seed = 2



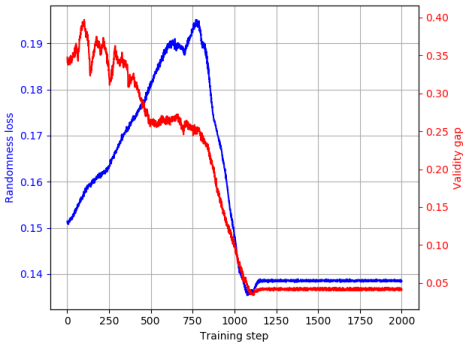
(a) Quadratic data - training losses.



(b) Simulated ambulance data - training losses



(c) Quadratic data - randomness loss and validity gap.

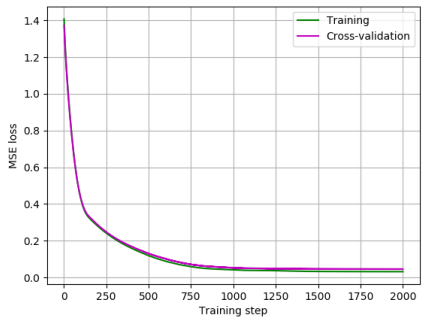


(d) Simulated ambulance data - randomness loss and validity gap.

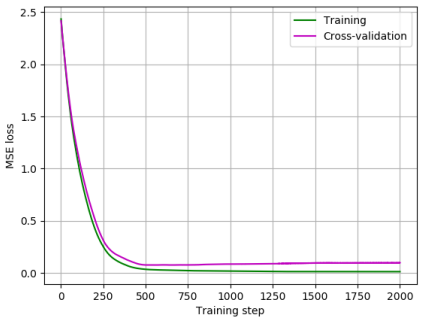
Figure 30: Evolution of training and 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 0 hidden layers. The quadratic data was generated with random seed 2 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 2 and had a window transform of length 4.

A.2.2 Neural networks with 1 hidden layer

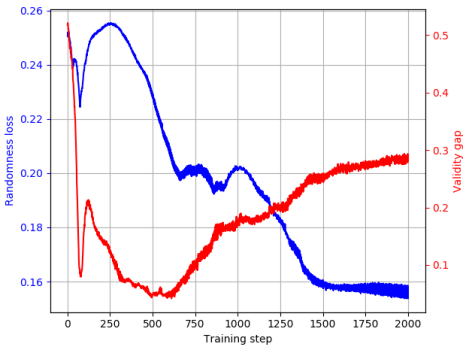
Random seed = 1



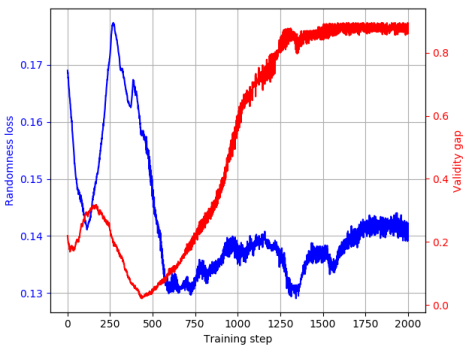
(a) Quadratic data.



(b) Simulated ambulance data.



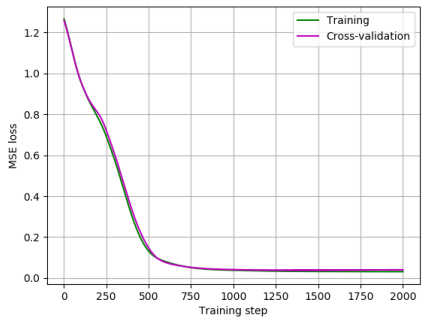
(c) Quadratic data.



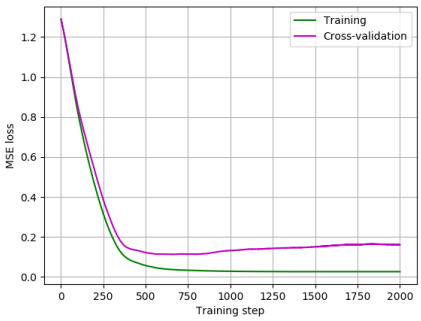
(d) Simulated ambulance data.

Figure 31: Evolution of training and 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 1 hidden layer. The quadratic data was generated with random seed 1 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 1 and had a window transform of length 4.

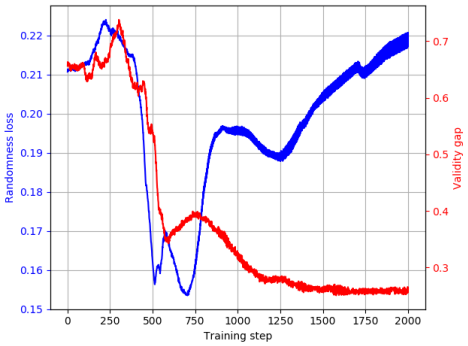
Random seed = 2



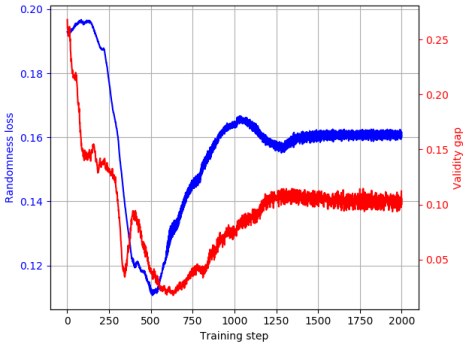
(a) Quadratic data.



(b) Simulated ambulance data.



(c) Quadratic data.

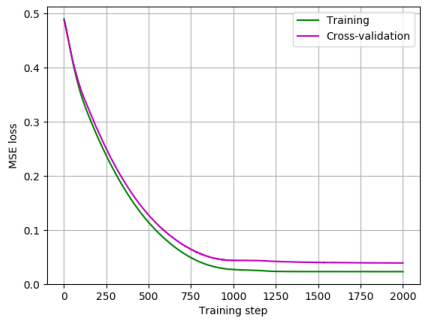


(d) Simulated ambulance data.

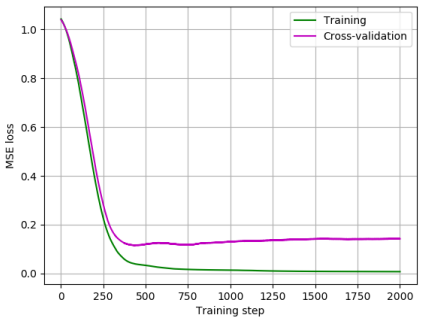
Figure 32: Evolution of training and 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 1 hidden layer. The quadratic data was generated with random seed 2 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 2 and had a window transform of length 4.

A.2.3 Neural networks with 2 hidden layers

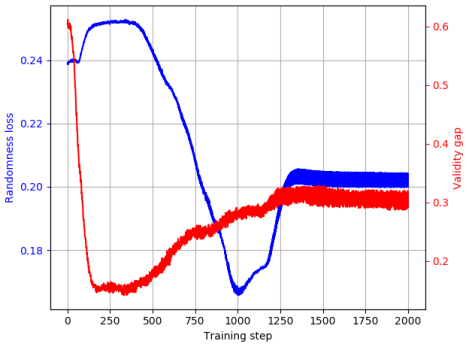
Random seed = 1



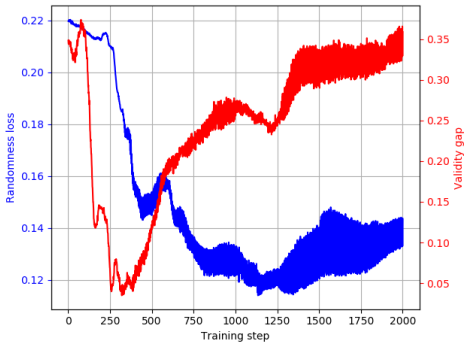
(a) Quadratic data.



(b) Simulated ambulance data.



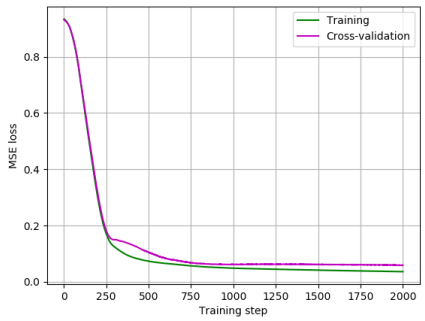
(c) Quadratic data.



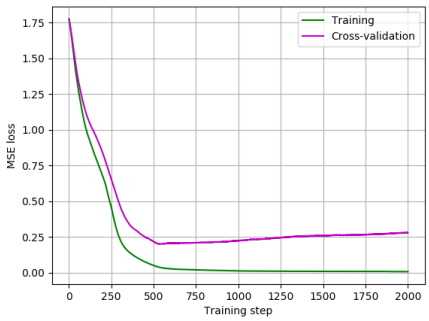
(d) Simulated ambulance data.

Figure 33: Evolution of training and 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 2 hidden layers. The quadratic data was generated with random seed 1 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 1 and had a window transform of length 4.

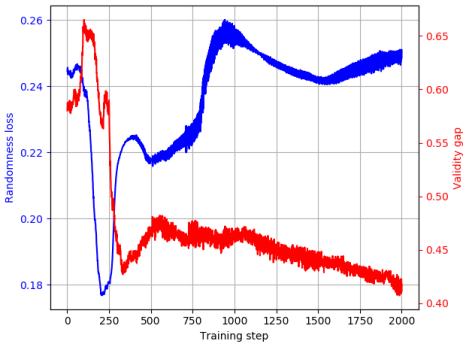
Random seed = 2



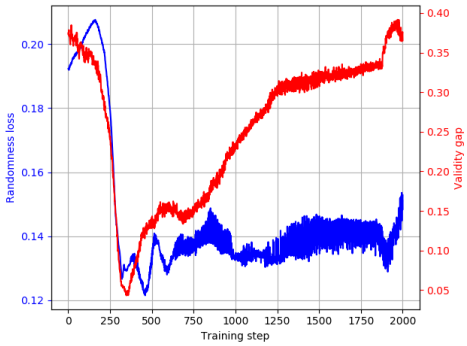
(a) Quadratic data.



(b) Simulated ambulance data.



(c) Quadratic data.



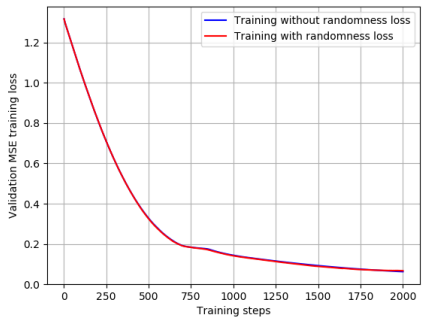
(d) Simulated ambulance data.

Figure 34: Evolution of training and 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 2 hidden layers. The quadratic data was generated with random seed 2 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 2 and had a window transform of length 4.

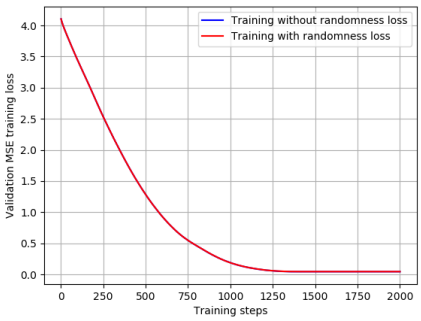
A.3 Experiment 4 Results

A.3.1 Neural networks with 0 hidden layers

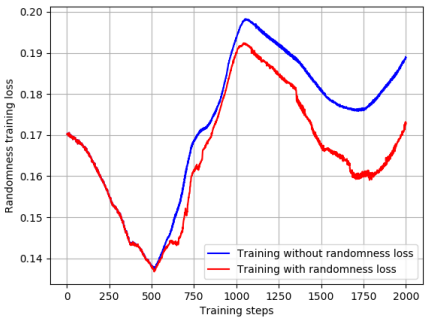
Random seed = 1



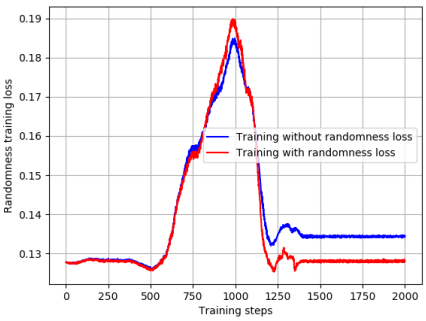
(a) Quadratic data.



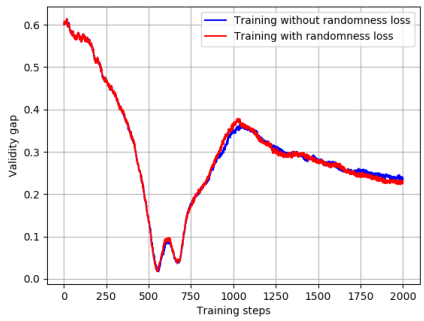
(b) Simulated ambulance data.



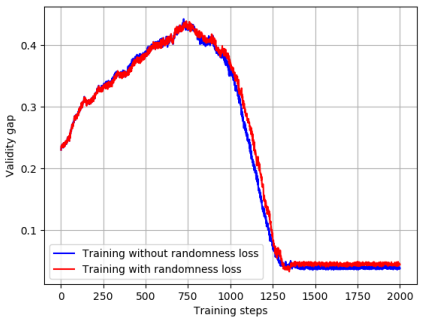
(c) Quadratic data.



(d) Simulated ambulance data.



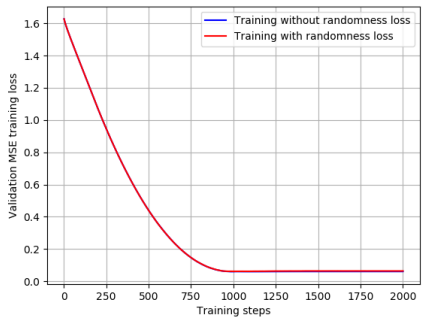
(e) Quadratic data.



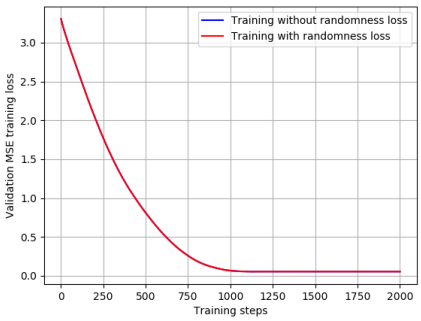
(f) Simulated ambulance data.

Figure 35: Comparison of the evolution of 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 0 hidden layers, with and without the use of randomness loss in the neural network's loss function. The quadratic data was generated with random seed 1 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 1 and had a window transform of length 4.

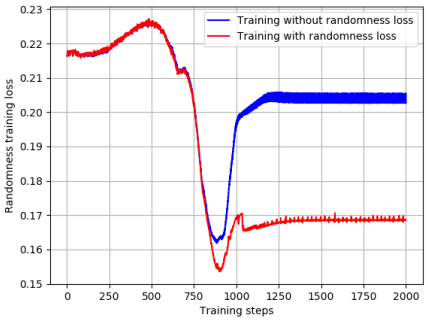
Random seed = 2



(a) Quadratic data.

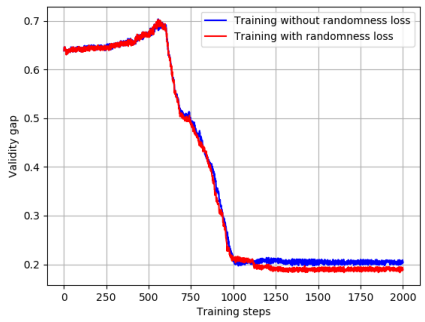


(b) Simulated ambulance data.

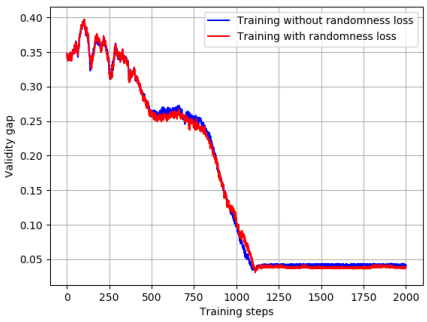


(c) Quadratic data.

(d) Simulated ambulance data.



(e) Quadratic data.

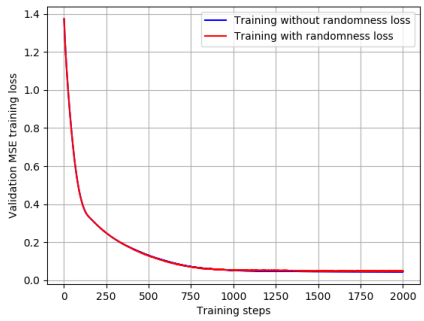


(f) Simulated ambulance data.

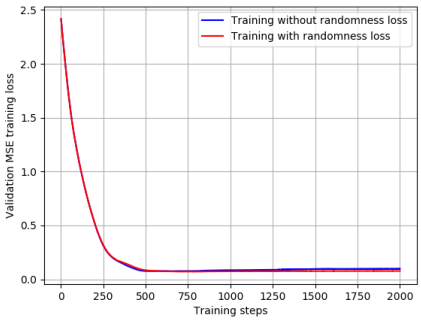
Figure 36: Comparison of the evolution of 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 0 hidden layers, with and without the use of randomness loss in the neural network's loss function. The quadratic data was generated with random seed 2 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 2 and had a window transform of length 4.

A.3.2 Neural networks with 1 hidden layer

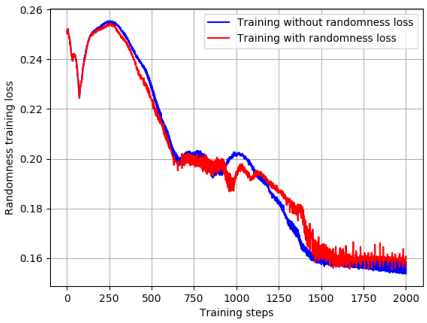
Random seed = 1



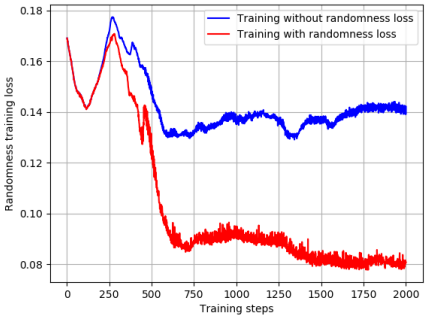
(a) Quadratic data.



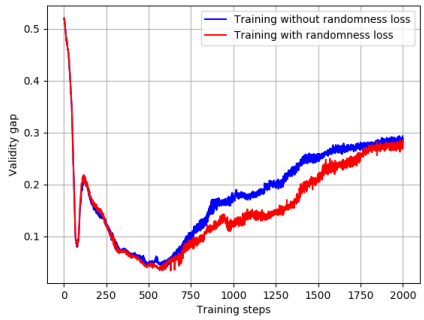
(b) Simulated ambulance data.



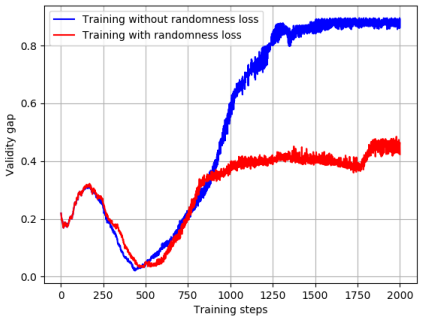
(c) Quadratic data.



(d) Simulated ambulance data.



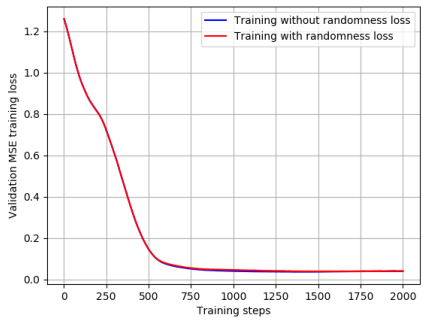
(e) Quadratic data.



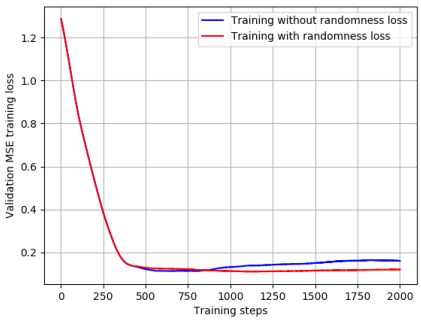
(f) Simulated ambulance data.

Figure 37: Comparison of the evolution of 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 1 hidden layer, with and without the use of randomness loss in the neural network's loss function. The quadratic data was generated with random seed 1 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 1 and had a window transform of length 4.

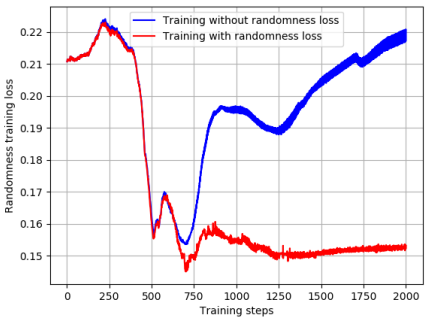
Random seed = 2



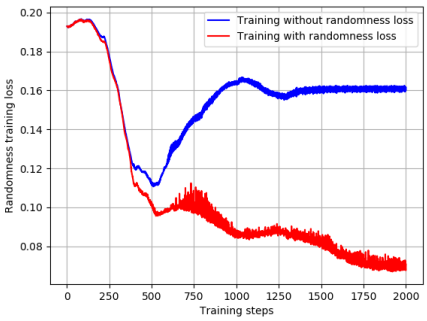
(a) Quadratic data.



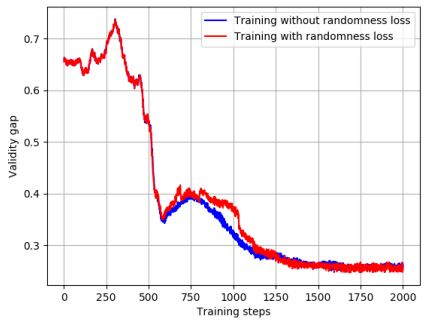
(b) Simulated ambulance data.



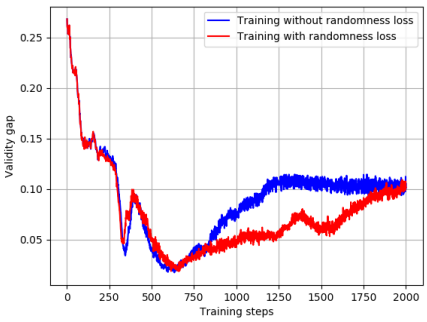
(c) Quadratic data.



(d) Simulated ambulance data.



(e) Quadratic data.

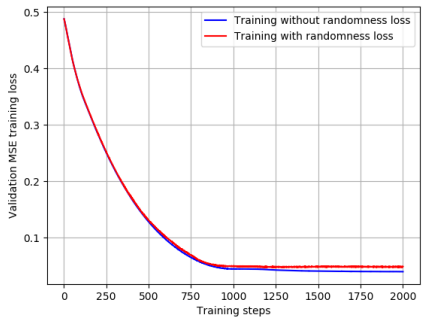


(f) Simulated ambulance data.

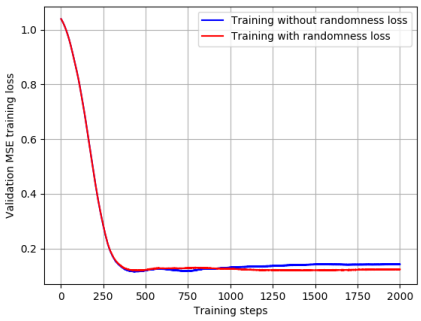
Figure 38: Comparison of the evolution of 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 1 hidden layer, with and without the use of randomness loss in the neural network's loss function. The quadratic data was generated with random seed 2 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 2 and had a window transform of length 4.

A.3.3 Neural networks with 2 hidden layers

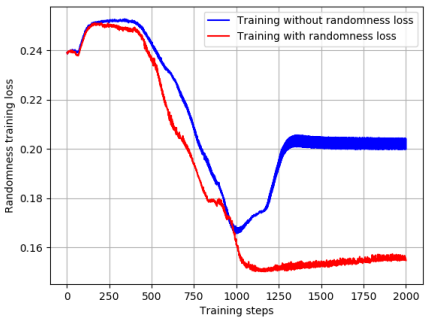
Random seed = 1



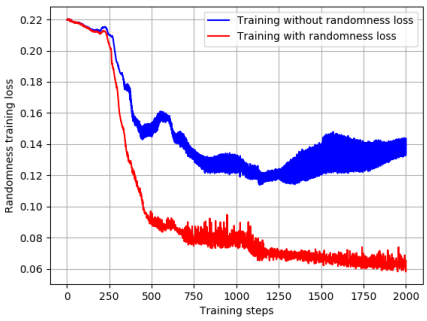
(a) Quadratic data.



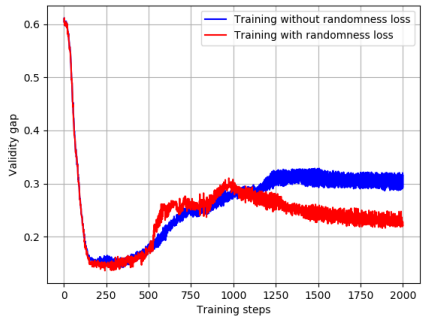
(b) Simulated ambulance data.



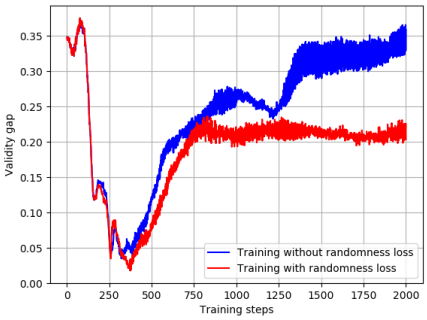
(c) Quadratic data.



(d) Simulated ambulance data.



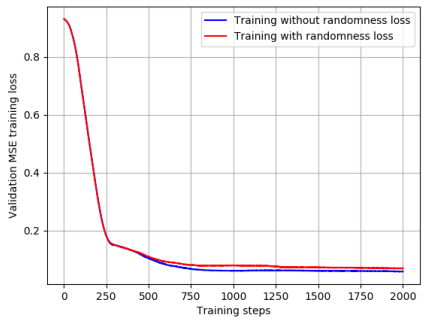
(e) Quadratic data.



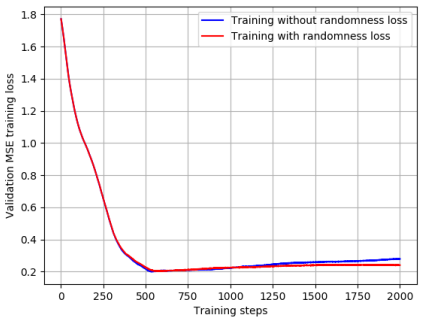
(f) Simulated ambulance data.

Figure 39: Comparison of the evolution of 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 2 hidden layers, with and without the use of randomness loss in the neural network's loss function. The quadratic data was generated with random seed 1 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 1 and had a window transform of length 4.

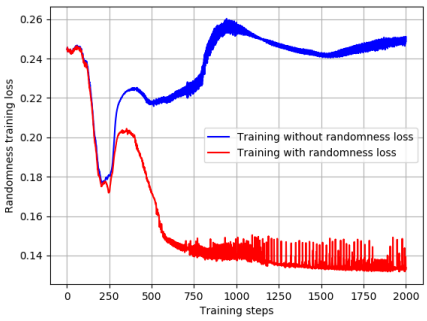
Random seed = 2



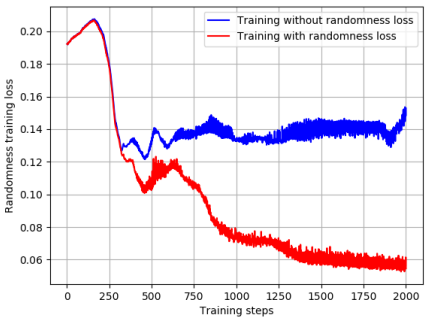
(a) Quadratic data.



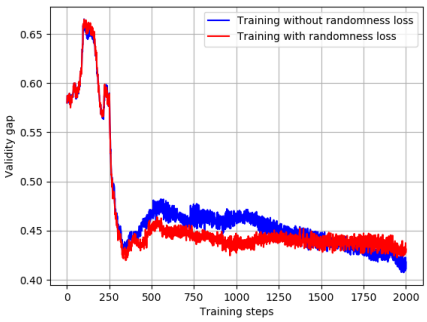
(b) Simulated ambulance data.



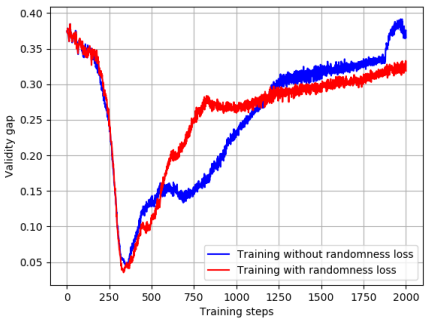
(c) Quadratic data.



(d) Simulated ambulance data.



(e) Quadratic data.



(f) Simulated ambulance data.

Figure 40: Comparison of the evolution of 5-fold cross-validation mean square error (MSE) loss, randomness loss (see Section 6.3.4) and validity gap (see Section 6.3.5) during the training process of a neural network with 2 hidden layers, with and without the use of randomness loss in the neural network's loss function. The quadratic data was generated with random seed 2 and had a window transform (see Section 6.3.3) of length 2. The simulated ambulance data was generated with random seed 2 and had a window transform of length 4.

B Simulated ambulance data

Dataset

100 training examples were generated using indices $i = 1, ..., 100$ and 100 test examples were generated all using index $i = 101$. The x_i were generated to simulate the hourly temperature, with rule

$$x_i = \alpha_0 + \alpha_1 sin(\alpha_2 + \alpha_3 i) + \alpha_4 sin(\alpha_5 + \alpha_6 i) + \gamma_i, \tag{12}$$

where the α_j terms are constant coefficients. The two sinusoids allow for simulation of a daily temperature cycle and an annual temperature cycle, and α_2 and α_5 allow the phasing of the sinusoids to be changed independently. The γ_i are iid Gaussian noise terms, with fixed parameters μ_x and σ_x.

The y_i labels are generated to simulate hourly ambulance demand, with rule

$$y_i = \beta_0 + \beta_1 i + \beta_2 sin(\beta_3 + \beta_4 i) + \beta_5 sin(\beta_6 + \beta_7 i) + \beta_8 x_i + \epsilon_i, \tag{13}$$

where the β terms are constant coefficients. The $\beta_8 x_i$ term is the simulated contribution of the hourly temperature to the ambulance demand. The β_0 term represents the baseline ambulance demand. $\beta_1 i$ represents the growth in demand over time, for example due to population growth or demographic changes. The two sinusoids allow daily and weekly ambulance demand cycles to be simulated. β_3 and β_6 allow these sinusoids to be phased independently. The ϵ_i are iid Gaussian noise terms, with fixed parameters μ_x and σ_x. σ_x is set to 5 to give an additional source of variation in ambulance demand that renders extreme fluctuations unlikely.

The coefficient values were generated according to Table 8. The seed of python's pseudo-random number generator was set before generating the random components.

Coefficient	Formula	Comment
α_0	$8 + Unif[-1, 1]$	Using 8C as average temperature
α_1	$7 + Unif[-1, 1]$	Using ± 7C as daily temperature swing
α_2	$Unif[0, 2\pi]$	Random phase offset
α_3	$2\pi/24$	24h in a day
α_4	$10 + Unif[-1, 1]$	Using 10C as seasonal temperature swing
α_5	$Unif[0, 2\pi]$	Random phase offset
α_6	$2\pi/(24 \times 365)$	24×365 hours in a year
μ_x	0	Standard normal
σ_x	1	Standard normal
β_0	$100 + Unif[-10, 10]$	Baseline ambulance demand
β_1	$5/(24 * 365)$	Gives increase of 5 over year
β_2	$30 + Unif[-2, 2]$	Using ± 30 as daily demand swing
β_3	$Unif[0, 2\pi]$	Random phase offset
β_4	$2\pi/24$	24h in a day
β_5	$20 + Unif[-2, 2]$	Using ± 20 as weekly demand swing
β_6	$Unif[0, 2\pi]$	Random phase offset
β_7	$2\pi/(24 \times 7)$	24×7 hours in a week
β_8	-1	Simplified effect of weather
μ_y	0	Zero mean additive noise
σ_y	5	Additional demand fluctuation

Table 8: Experiment 3 coefficients

C Simulated estimation of p-value randomness scores

A set of $n = 100$ simulated p-values,

$$\left\{\frac{0+\epsilon}{n}, \frac{1+\epsilon}{n}, ..., \frac{(n-1)+\epsilon}{n}\right\}, \tag{14}$$

was generated, where n is the number of datapoints and each ϵ is an independent sample from a uniform random variable in [0, 1]. The number of p-values generated matches the number of training data points in the relevant experiment. This is an efficient method of generating the p-values (albeit in a highly unlikely order) that a conformal transducer would generate from a set of n unique nonconformity scores according to (4). Because $\epsilon \in [0, 1]$, each of the p-values is in [0, 1]. The simulated p-values were then shuffled, and the $p_{dev,3}$, $p_{dev,5}$, $p_{dev,7}$ and $p_{dev,9}$ and final martingale values were calculated using the shuffled values. This process was repeated 10,000 times, setting the seed of python's pseudo-random number generator to 0 at the start but not between iterations. For each quantity the mean and RMS deviation from the mean was calculated.

$\boldsymbol{p_{dev,3}}$	$\boldsymbol{p_{dev,5}}$	$\boldsymbol{p_{dev,7}}$	$\boldsymbol{p_{dev,9}}$
0.134 ± 0.011	0.102 ± 0.012	0.085 ± 0.013	0.074 ± 0.013

Table 9: $\boldsymbol{p_{dev,3}}$, $\boldsymbol{p_{dev,5}}$, $\boldsymbol{p_{dev,7}}$ and $\boldsymbol{p_{dev,9}}$ values, as detailed in Section 6.3.4, averaged over 10000 simulated runs.

D Software packages

Package	Version
python	3.6.2
matplotlib	2.0.2
numpy	1.13.1
scipy	0.19.1
scikit-learn	0.18.2
tensorflow	1.2.1

Table 10: Software packages and versions used for the experiments in this report.

References

[1] Blanquart JP, et al. Criticality categories across safety standards in different domains; 2012. Presented at Embedded Real-Time Software and Systems Conference 2012, Toulouse, France.

[2] International Electrotechnical Commission. Functional Safety and IEC 61508; 2010. Date not given. [Accessed 18 June 2017]. `http://www.iec.ch/functionalsafety/`.

[3] Christoffersen P. Elements of Financial Risk Management. Waltham, MA: Elsevier; 2012.

[4] Vovk V, Gammerman A, Shafer G. Algorithmic Learning in a Random World. New York: Springer; 2005. Theorem 8.1, p.193.

[5] Vovk V, Gammerman A, Saunders C. Machine-Learning Applications of Algorithmic Randomness. In: In Proceedings of the Sixteenth International Conference on Machine Learning. Morgan Kaufmann; 1999. p. 444–453.

[6] Dashevskiy M, Luo Z. Time series prediction with performance guarantee. IET communications. 2011;5(8):1044–1051.

[7] Szegedy C, Ioffe S, Vanhoucke V, Alemi AA. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In: ICLR 2016 Workshop; 2016. Available from: `https://arxiv.org/abs/1602.07261`.

[8] Johnson M, Schuster M, Le QV, Krikun M, Wu Y, Chen Z, et al. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. CoRR. 2016;abs/1611.04558. Available from: `http://arxiv.org/abs/1611.04558`.

[9] Vapnik VN. Statistical Learning Theory. New York, NY: Wiley; 1998.

[10] von Luxburg U, Schoelkopf B. Statistical Learning Theory: Models, Concepts, and Results. ArXiv e-prints. 2008 Oct;v1.

[11] Vovk V, Gammerman A, Shafer G. Algorithmic Learning in a Random World. New York: Springer; 2005. Section 1.2, p.3–4.

[12] Shawe-Taylor J, Bartlett PL, Williamson RC, Anthony M. Structural risk minimization over data-dependent hierarchies. IEEE Transactions on Information Theory. 1998 Sep;44(5):1926–1940.

[13] Shawe-Taylor J. Classification Accuracy Based on Observed Margin. Algorithmica. 1998 Sep;22(1):157–172. Available from: `https://doi.org/10.1007/PL00013827`.

[14] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation. 1997 Apr;1(1):67–82.

[15] Rice JA. Mathematical Statistics and Data Analysis. Duxbury; 2007.

[16] Fisher RA. The Fiducial Argument in Statistical Inference. Annals of Eugenics. 1935;6:391–398. Available from: https://digital.library.adelaide.edu.au/dspace/bitstream/2440/15222/1/125.pdf.

[17] Shafer G, Vovk V. A Tutorial on Conformal Prediction. Journal of Machine Learning Research. 2008;9:371–421. Available from: http://jmlr.csail.mit.edu/papers/volume9/shafer08a/shafer08a.pdf.

[18] Li M, Vitányi P. An introduction to Kolmogorov complexity and its applications. (Second edition). Computers & Mathematics with Applications. 1997;34(10):137. Available from: http://www.sciencedirect.com/science/article/pii/S0898122197902133.

[19] Cowles MK. Applied Bayesian Statistics. Springer-Verlag New York; 2013.

[20] Melluish T, Saunders C, Nouretdinov I, Vovk V. Comparing the Bayes and Typicalness Frameworks. In: Proceedings of the 12th European Conference on Machine Learning. EMCL '01. London, UK, UK: Springer-Verlag; 2001. p. 360–371. Available from: http://dl.acm.org/citation.cfm?id=645328.650016.

[21] Bolstad WM. Introduction to Bayesian Statistics. Hoboken, NJ: Wiley; 2007.

[22] Cesa-Bianchi N, Lugosi G. Prediction, Learning, and Games. New York, NY, USA: Cambridge University Press; 2006.

[23] Khosravi A, Nahavandi S, Creighton D, Atiya AF. Comprehensive Review of Neural Network-Based Prediction Intervals and New Advances. IEEE Transactions on Neural Networks. 2011 Sept;22(9):1341–1356.

[24] Hwang JTG, Ding AA. Prediction Intervals for Artificial Neural Networks. Journal of the American Statistical Association. 1997;92(438):748–757. Available from: http://dx.doi.org/10.1080/01621459.1997.10474027.

[25] Vleaux RDD, Schumi J, Schweinsberg J, Ungar LH. Prediction Intervals for Neural Networks via Nonlinear Regression. Technometrics. 1998;40(4):273–282. Available from: http://dx.doi.org/10.1080/00401706.1998.10485556.

[26] Heskes T. Practical Confidence and Prediction Intervals. In: Mozer MC, Jordan MI, Petsche T, editors. Advances in Neural Information Processing Systems 9. MIT Press; 1997. p. 176–182. Available from: http://papers.nips.cc/paper/1306-practical-confidence-and-prediction-intervals.pdf.

[27] Khosravi A, Nahavandi S, Creighton D. Quantifying uncertainties of neural network-based electricity price forecasts. Applied Energy. 2013;112:120 – 129. Available from: http://www.sciencedirect.com/science/article/pii/S0306261913004881.

[28] Bishop CM. Pattern Recognition and Machine Learning. Springer; 2006.

[29] Carlsson L, Eklund M, Norinder U. In: Iliadis L, Maglogiannis I, Papadopoulos H, Sioutas S, Makris C, editors. Aggregated Conformal Prediction. Berlin, Heidelberg: Springer Berlin Heidelberg; 2014. p. 231–240. Available from: https://doi.org/10.1007/978-3-662-44722-2_25.

[30] Norinder U, Carlsson L, Boyer S, Eklund M. Introducing conformal prediction in predictive modeling for regulatory purposes. A transparent and flexible alternative to applicability domain determination. Regulatory Toxicology and Pharmacology. 2015;71(2):279 – 284. Available from: http://www.sciencedirect.com/science/article/pii/S0273230014003432.

[31] Norinder U, Rybacka A, Andersson PL. Conformal prediction to define applicability domain – A case study on predicting ER and AR binding. SAR and QSAR in Environmental Research. 2016;27(4):303–316. PMID: 27088868. Available from: http://dx.doi.org/10.1080/1062936X.2016.1172665.

[32] Norinder U, Boyer S. Binary classification of imbalanced datasets using conformal prediction. Journal of Molecular Graphics and Modelling. 2017;72:256 – 265. Available from: http://www.sciencedirect.com/science/article/pii/S1093326316301644.

[33] Devetyarov D, Nouretdinov I, Burford B, Camuzeaux S, Gentry-Maharaj A, Tiss A, et al. Conformal predictors in early diagnostics of ovarian and breast cancers. Progress in Artificial Intelligence. 2012 Sep;1(3):245–257. Available from: https://doi.org/10.1007/s13748-012-0021-y.

[34] Alex G, Ilia N, Brian B, Alexey C, Vladimir V, Zhiyuan L. Clinical Mass Spectrometry Proteomic Diagnosis by Conformal Predictors. Statistical Applications in Genetics and Molecular Biology. 2008;7(2):1–12. Available from: http://EconPapers.repec.org/RePEc:bpj:sagmbi:v:7:y:2008:i:2:n:13.

[35] Papadopoulos H, Gammerman A, Vovk V. Reliable diagnosis of acute abdominal pain with conformal prediction. International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications. 2009;17(2-3):127–137.

[36] Nouretdinov I, Costafreda SG, Gammerman A, Chervonenkis A, Vovk V, Vapnik V, et al. Machine learning classification with confidence: application of transductive conformal predictors to MRI-based diagnostic and prognostic markers in depression. Neuroimage. 2011 May;56(2):809–813.

[37] Lambrou A, Papadopoulos H, Kyriacou E, Pattichis CS, Pattichis MS, Gammerman A, et al. In: Papadopoulos H, Andreou AS, Bramer M, editors. Assessment of Stroke Risk Based on Morphological Ultrasound Image Analysis with Conformal Prediction. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010. p. 146–153. Available from: https://doi.org/10.1007/978-3-642-16239-8_21.

[38] Vega J, Murari A, Dormido-Canto S, Moreno R, Pereira A, Acero A, et al. Adaptive high learning rate probabilistic disruption predictors from scratch

for the next generation of tokamaks. Nuclear Fusion. 2014;54(12):123001. Available from: http://stacks.iop.org/0029-5515/54/i=12/a=123001.

[39] Nouretdinov I, Li G, Gammerman A, Luo Z. Application of Conformal Predictors to Tea Classification Based on Electronic Nose; 2010.

[40] Wang Z, Sun X, Miao J, Wang Y, Luo Z, Li G. Conformal Prediction Based on K-Nearest Neighbors for Discrimination of Ginsengs by a Home-Made Electronic Nose. 2017 08;17:1869.

[41] Balasubramanian VN, Ho SS, Vovk V. Conformal Prediction for Reliable Machine Learning. Waltham, MA: Elsevier; 2014. Eq.(1.20), p.15–16.

[42] Nouretdinov I, Melluish T, Vovk V. Ridge Regression Confidence Machine. In: Proceedings of the Eighteenth International Conference on Machine Learning. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2001. p. 385–392. Available from: http://dl.acm.org/citation.cfm?id=645530.655664.

[43] Neyman J. Outline of a Theory of Statistical Estimation Based on the Classical Theory of Probability. Philosophical Transactions of the Royal Society of London Series A, Mathematical and Physical Sciences. 1937;236(767):333–380. Available from: http://www.jstor.org/stable/91337.

[44] Balasubramanian VN, Ho SS, Vovk V. Conformal Prediction for Reliable Machine Learning. Waltham, MA: Elsevier; 2014. Chapter 12, p.240–241.

[45] Fedorova V, Gammerman A, Nouretdinov I, Vovk V. Plug-in martingales for testing exchangeability on-line. CoRR. 2012;abs/1204.3251. Available from: http://arxiv.org/abs/1204.3251.

[46] Fedorova V. Conformal Prediction and Testing under On-line Compression Models. Royal Holloway University of London; 2014.

[47] Vovk V, Gammerman A, Shafer G. Algorithmic Learning in a Random World. New York: Springer; 2005. Eq.(2.20), p.27.

[48] Kessy A, Lewin A, Strimmer K. Optimal whitening and decorrelation. ArXiv e-prints. 2015 Dec;v4.

[49] He K, Zhang X, Ren S, Sun J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. CoRR. 2015;abs/1502.01852. Available from: http://arxiv.org/abs/1502.01852.