[*Read before* The Royal Statistical Society *at the Society's 2023 annual conference held in* Harrogate *on Wednesday, September 6th, 2023,* the President, Dr Andrew Garrett, *in the Chair*.]

# Automatic Change-Point Detection in Time Series via Deep Learning

Jie Li†
*Department of Statistics, London School of Economics and Political Science, London, UK*
E-mail: j.li196@lse.ac.uk
Paul Fearnhead
*Department of Mathematics and Statistics, Lancaster University, Lancaster, UK*
E-mail: p.fearnhead@lancaster.ac.uk
Piotr Fryzlewicz
*Department of Statistics, London School of Economics and Political Science, London, UK*
E-mail: p.fryzlewicz@lse.ac.uk
Tengyao Wang
*Department of Statistics, London School of Economics and Political Science, London, UK*
E-mail: t.wang59@lse.ac.uk

**Summary**. Detecting change-points in data is challenging because of the range of possible types of change and types of behaviour of data when there is no change. Statistically efficient methods for detecting a change will depend on both of these features, and it can be difficult for a practitioner to develop an appropriate detection method for their application of interest. We show how to automatically generate new offline detection methods based on training a neural network. Our approach is motivated by many existing tests for the presence of a change-point being representable by a simple neural network, and thus a neural network trained with sufficient data should have performance at least as good as these methods. We present theory that quantifies the error rate for such an approach, and how it depends on the amount of training data. Empirical results show that, even with limited training data, its performance is competitive with the standard CUSUM-based classifier for detecting a change in mean when the noise is independent and Gaussian, and can substantially outperform it in the presence of auto-correlated or heavy-tailed noise. Our method also shows strong results in detecting and localising changes in activity based on accelerometer data.

*Keywords*: Automatic statistician; Classification; Likelihood-free inference; Neural networks; Structural breaks; Supervised learning

## 1. Introduction

Detecting change-points in data sequences is of interest in many application areas such as bioinformatics (Picard et al., 2005), climatology (Reeves et al., 2007), signal process-

†Addresses for correspondence: Jie Li, Department of Statistics, London School of Economics and Political Science, London, WC2A 2AE. **Email**: j.li196@lse.ac.uk

ing (Haynes et al., 2017) and neuroscience (Oh et al., 2005). In this work, we are primarily concerned with the problem of offline change-point detection, where the entire data is available to the analyst beforehand. Over the past few decades, various methodologies have been extensively studied in this area, see Killick et al. (2012); Jandhyala et al. (2013); Fryzlewicz (2014, 2023); Wang and Samworth (2018); Truong et al. (2020) and references therein. Most research on change-point detection has concentrated on detecting and localising different types of change, e.g. change in mean (Killick et al., 2012; Fryzlewicz, 2014), variance (Gao et al., 2019; Li et al., 2015), median (Fryzlewicz, 2021) or slope (Baranowski et al., 2019; Fearnhead et al., 2019), amongst many others.

Many change-point detection methods are based upon modelling data when there is no change and when there is a single change, and then constructing an appropriate test statistic to detect the presence of a change (e.g. James et al., 1987; Fearnhead and Rigaill, 2020). The form of a good test statistic will vary with our modelling assumptions and the type of change we wish to detect. This can lead to difficulties in practice. As we use new models, it is unlikely that there will be a change-point detection method specifically designed for our modelling assumptions. Furthermore, developing an appropriate method under a complex model may be challenging, while in some applications an appropriate model for the data may be unclear but we may have substantial historical data that shows what patterns of data to expect when there is, or is not, a change.

In these scenarios, currently a practitioner would need to choose the existing change detection method which seems the most appropriate for the type of data they have and the type of change they wish to detect. To obtain reliable performance, they would then need to adapt its implementation, for example tuning the choice of threshold for detecting a change. Often, this would involve applying the method to simulated or historical data.

To address the challenge of automatically developing new change detection methods, this paper is motivated by the question: Can we construct new test statistics for detecting a change based only on having labelled examples of change-points? We show that this is indeed possible by training a neural network to classify whether or not a data set has a change of interest. This turns change-point detection in a supervised learning problem.

A key motivation for our approach are results that show many common test statistics for detecting changes, such as the CUSUM test for detecting a change in mean, can be represented by simple neural networks. This means that with sufficient training data, the classifier learnt by such a neural network will give performance at least as good as classifiers corresponding to these standard tests. In scenarios where a standard test, such as CUSUM, is being applied but its modelling assumptions do not hold, we can expect the classifier learnt by the neural network to outperform it.

There has been increasing recent interest in whether ideas from machine learning, and methods for classification, can be used for change-point detection. Within computer science and engineering, these include a number of methods designed for and that show promise on specific applications (e.g. Ahmadzadeh, 2018; De Ryck et al., 2021; Gupta et al., 2022; Huang et al., 2023). Within statistics, Londschien et al. (2022) and Lee et al. (2023) consider training a classifier as a way to estimate the likelihood-ratio statistic for a change. However these methods train the classifier in an un-supervised way on the data being analysed, using the idea that a classifier would more easily distinguish between two segments of data if they are separated by a change-point. Chang et al. (2019) use

simulated data to help tune a kernel-based change detection method. Methods that use historical, labelled data have been used to train the tuning parameters of change-point algorithms (e.g. Hocking et al., 2015; Liehrmann et al., 2021). Also, neural networks have been employed to construct similarity scores of new observations to learned pre-change distributions for online change-point detection (Lee et al., 2023). However, we are unaware of any previous work using historical, labelled data to develop offline change-point methods. As such, and for simplicity, we focus on the most fundamental aspect, namely the problem of detecting a single change. Detecting and localising multiple changes is considered in Section 6 when analysing activity data. We remark that by viewing the change-point detection problem as a classification instead of a testing problem, we aim to control the overall misclassification error rate instead of handling the Type I and Type II errors separately. In practice, asymmetric treatment of the two error types can be achieved by suitably re-weighting misclassification in the two directions in the training loss function.

The method we develop has parallels with likelihood-free inference methods (Gourieroux et al., 1993; Beaumont, 2019) in that one application of our work is to use the ability to simulate from a model so as to circumvent the need to analytically calculate likelihoods. However, the approach we take is very different from standard likelihood-free methods which tend to use simulation to estimate the likelihood function itself. By comparison, we directly target learning a function of the data that can discriminate between instances that do or do not contain a change (though see Gutmann et al., 2018, for likelihood-free methods based on re-casting the likelihood as a classification problem).

For an introduction to the statistical aspects of neural network-based classification, albeit not specifically in a change-point context, see Ripley (1994).

We now briefly introduce our notation. For any $n \in \mathbb{Z}^+$, we define $[n] := \{1, \ldots, n\}$. We take all vectors to be column vectors unless otherwise stated. Let $\mathbf{1}_n$ be the all-one vector of length $n$. Let $\mathbb{1}\{\cdot\}$ represent the indicator function. The vertical symbol $|\cdot|$ represents the absolute value or cardinality of $\cdot$ depending on the context. For vector $\boldsymbol{x} = (x_1, \ldots, x_n)^\top$, we define its $p$-norm as $\|\boldsymbol{x}\|_p := \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}, p \geq 1$; when $p = \infty$, define $\|\boldsymbol{x}\|_\infty := \max_i |x_i|$. All proofs, as well as additional simulations and real data analyses appear in the supplement.

## 2. Neural networks

The initial focus of our work is on the binary classification problem for whether a change-point exists in a given time series. We will work with multilayer neural networks with Rectified Linear Unit (ReLU) activation functions and binary output. The multilayer neural network consists of an input layer, hidden layers and an output layer, and can be represented by a directed acyclic graph, see Figure 1. Let $L \in \mathbb{Z}^+$ represent the number of hidden layers and $\boldsymbol{m} = (m_1, \ldots, m_L)^\top$ the vector of the hidden layers widths, i.e. $m_i$ is the number of nodes in the $i$th hidden layer. For a neural network with $L$ hidden layers we use the convention that $m_0 = n$ and $m_{L+1} = 1$. For any bias vector $\boldsymbol{b} = (b_1, b_2, \ldots, b_r)^\top \in \mathbb{R}^r$, define the shifted activation function $\sigma_{\boldsymbol{b}} : \mathbb{R}^r \to \mathbb{R}^r$:

$$\sigma_{\boldsymbol{b}}((y_1, \ldots, y_r)^\top) = (\sigma(y_1 - b_1), \ldots, \sigma(y_r - b_r))^\top,$$

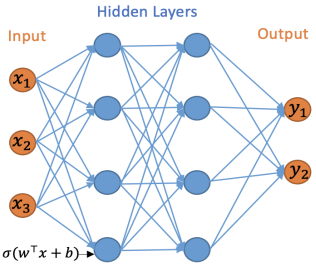



**Fig. 1.** A neural network with 2 hidden layers and width vector $\mathbf{m} = (4, 4)$.

where $\sigma(x) = \max(x, 0)$ is the ReLU activation function. The neural network can be mathematically represented by the composite function $h : \mathbb{R}^n \to \{0, 1\}$ as

$$h(\boldsymbol{x}) \coloneqq \sigma_\lambda^* W_L \sigma_{\boldsymbol{b}_L} W_{L-1} \sigma_{\boldsymbol{b}_{L-1}} \cdots W_1 \sigma_{\boldsymbol{b}_1} W_0 \boldsymbol{x}, \tag{1}$$

where $\sigma_\lambda^*(x) = \mathbb{1}\{x > \lambda\}$, $\lambda > 0$ and $W_\ell \in \mathbb{R}^{m_{\ell+1} \times m_\ell}$ for $\ell \in \{0, \ldots, L\}$ represent the weight matrices. We define the function class $\mathcal{H}_{L,\boldsymbol{m}}$ to be the class of functions $h(\boldsymbol{x})$ with $L$ hidden layers and width vector $\boldsymbol{m}$.

The output layer in (1) employs the shifted heaviside function $\sigma_\lambda^*(x)$, which is used for binary classification as the final activation function. This choice is guided by the fact that we use the 0-1 loss, which focuses on the percentage of samples assigned to the correct class, a natural performance criterion for binary classification. Besides its wide adoption in machine learning practice, another advantage of using the 0-1 loss is that it is possible to utilise the theory of the Vapnik–Chervonenkis (VC) dimension (see, e.g. Shalev-Shwartz and Ben-David, 2014, Definition 6.5) to bound the generalisation error of a binary classifier equipped with this loss; indeed, this is the approach we take in this work. The relevant results regarding the VC dimension of neural network classifiers are e.g. in Bartlett et al. (2019). As in Schmidt-Hieber (2020), we work with the exact minimiser of the empirical risk. In both binary or multiclass classification, it is possible to work with other losses which make it computationally easier to minimise the corresponding risk, see e.g. Bos and Schmidt-Hieber (2022), who use a version of the cross-entropy loss. However, loss functions different from the 0-1 loss make it impossible to use VC-dimension arguments to control the generalisation error, and more involved arguments, such as those using the covering number (Bos and Schmidt-Hieber, 2022) need to be used instead. We do not pursue these generalisations in the current work.

## 3. CUSUM-based classifier and its generalisations are neural networks

### 3.1. Change in mean

We initially consider the case of a single change-point with an unknown location $\tau \in [n-1]$, $n \geq 2$, in the model

$$\begin{aligned} \boldsymbol{X} &= \boldsymbol{\mu} + \boldsymbol{\xi}, \\ \boldsymbol{\mu} &= (\mu_{\mathrm{L}} \mathbb{1}\{i \leq \tau\} + \mu_{\mathrm{R}} \mathbb{1}\{i > \tau\})_{i \in [n]} \in \mathbb{R}^n, \end{aligned}$$

where $\mu_{\mathrm{L}}, \mu_{\mathrm{R}}$ are the unknown signal values before and after the change-point; $\boldsymbol{\xi} \sim N_n(0, I_n)$. The CUSUM test is widely used to detect mean changes in univariate data. For the observation $\boldsymbol{x}$, the CUSUM transformation $\mathcal{C} : \mathbb{R}^n \to \mathbb{R}^{n-1}$ is defined as $\mathcal{C}(\boldsymbol{x}) := (\boldsymbol{v}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{v}_{n-1}^\top \boldsymbol{x})^\top$, where $\boldsymbol{v}_i := \left(\sqrt{\frac{n-i}{in}}\mathbf{1}_i^\top, -\sqrt{\frac{i}{(n-i)n}}\mathbf{1}_{n-i}^\top\right)^\top$ for $i \in [n-1]$. Here, for each $i \in [n-1]$, $(\boldsymbol{v}_i^\top \boldsymbol{x})^2$ is the log likelihood-ratio statistic for testing a change at time $i$ against the null of no change (e.g. Baranowski et al., 2019). For a given threshold $\lambda > 0$, the classical CUSUM test for a change in the mean of the data is defined as

$$h_\lambda^{\mathrm{CUSUM}}(\boldsymbol{x}) = \mathbb{1}\{\|\mathcal{C}(\boldsymbol{x})\|_\infty > \lambda\}.$$

The following lemma shows that $h_\lambda^{\mathrm{CUSUM}}(\boldsymbol{x})$ can be represented as a neural network.

LEMMA 3.1. *For any $\lambda > 0$, we have $h_\lambda^{\mathrm{CUSUM}}(\boldsymbol{x}) \in \mathcal{H}_{1,2n-2}$.*

The fact that the widely-used CUSUM statistic can be viewed as a simple neural network has far-reaching consequences: this means that given enough training data, a neural network architecture that permits the CUSUM-based classifier as its special case cannot do worse than CUSUM in classifying change-point versus no-change-point signals. This serves as the main motivation for our work, and a prelude to our next results.

### 3.2. Beyond the mean change model

We can generalise the simple change in mean model to allow for different types of change or for non-independent noise. In this section, we consider change-point models that can be expressed as a change in regression problem, where the model for data given a change at $\tau$ is of the form

$$\boldsymbol{X} = \boldsymbol{Z}\boldsymbol{\beta} + \boldsymbol{c}_\tau \phi + \boldsymbol{\Gamma}\boldsymbol{\xi}, \tag{2}$$

where for some $p \geq 1$, $\boldsymbol{Z}$ is an $n \times p$ matrix of covariates for the model with no change, $\boldsymbol{c}_\tau$ is an $n \times 1$ vector of covariates specific to the change at $\tau$, and the parameters $\boldsymbol{\beta}$ and $\phi$ are, respectively, a $p \times 1$ vector and a scalar. The noise is defined in terms of an $n \times n$ matrix $\boldsymbol{\Gamma}$ and an $n \times 1$ vector of independent standard normal random variables, $\boldsymbol{\xi}$.

For example, the change in mean problem has $p = 1$, with $\boldsymbol{Z}$ a column vector of ones, and $\boldsymbol{c}_\tau$ being a vector whose first $\tau$ entries are zeros, and the remaining entries are ones. In this formulation $\beta$ is the pre-change mean, and $\phi$ is the size of the change. The change in slope problem (Fearnhead et al., 2019) has $p = 2$ with the columns of $\boldsymbol{Z}$ being a vector of ones, and a vector whose $i$th entry is $i$; and $\boldsymbol{c}_\tau$ has $i$th entry that is $\max\{0, i - \tau\}$. In this formulation $\boldsymbol{\beta}$ defines the pre-change linear mean, and $\phi$ the size of the change in slope. Choosing $\boldsymbol{\Gamma}$ to be proportional to the identity matrix gives a model with independent, identically distributed noise; but other choices would allow for auto-correlation.

The following result is a generalisation of Lemma 3.1, which shows that the likelihood-ratio test for (2), viewed as a classifier, can be represented by our neural network.

LEMMA 3.2. *Consider the change-point model (2) with a possible change at $\tau \in [n-1]$. Assume further that $\boldsymbol{\Gamma}$ is invertible. Then there is an $h^* \in \mathcal{H}_{1,2n-2}$ equivalent to the likelihood-ratio test for testing $\phi = 0$ against $\phi \neq 0$.*

Importantly, this result shows that for this much wider class of change-point models, we can replicate the likelihood-ratio-based classifier for change using a simple neural network.

Other types of changes can be handled by suitably pre-transforming the data. For instance, squaring the input data would be helpful in detecting changes in the variance and if the data followed an AR(1) structure, then changes in autocorrelation could be handled by including transformations of the original input of the form $(x_t x_{t+1})_{t=1,\ldots,n-1}$. On the other hand, even if such transformations are not supplied as the input, a neural network of suitable depth is able to approximate these transformations and consequently successfully detect the change (Schmidt-Hieber, 2020, Lemma A.2). This is illustrated in Figure S3 of the supplementary material, where we compare the performance of neural network based classifiers of various depths constructed with and without using the transformed data as inputs.

## 4. Generalisation error of neural network change-point classifiers

In Section 3, we showed that CUSUM and generalised CUSUM could be represented by a neural network. Therefore, with a large enough amount of training data, a trained neural network classifier that included CUSUM, or generalised CUSUM, as a special case, would perform no worse than it on unseen data. In this section, we provide generalisation bounds for a neural network classifier for the change-in-mean problem, given a finite amount of training data. En route to this main result, stated in Theorem 4.3, we provide generalisation bounds for the CUSUM-based classifier, in which the threshold has been chosen on a finite training data set.

We write $P(n, \tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}})$ for the distribution of the multivariate normal random vector $\boldsymbol{X} \sim N_n(\boldsymbol{\mu}, I_n)$ where $\boldsymbol{\mu} := (\mu_{\mathrm{L}}\mathbb{1}\{i \leq \tau\} + \mu_{\mathrm{R}}\mathbb{1}\{i > \tau\})_{i\in[n]}$. Define $\eta := \tau/n$. Lemma 4.1 and Corollary 4.1 control the misclassification error of the CUSUM-based classifier.

LEMMA 4.1. *Fix $\varepsilon \in (0, 1)$. Suppose $\boldsymbol{X} \sim P(n, \tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}})$ for some $\tau \in \mathbb{Z}^+$ and $\mu_{\mathrm{L}}, \mu_{\mathrm{R}} \in \mathbb{R}$.*

*(a) If $\mu_{\mathrm{L}} = \mu_{\mathrm{R}}$, then $\mathbb{P}\{\|\mathcal{C}(\boldsymbol{X})\|_\infty > \sqrt{2\log(n/\varepsilon)}\} \leq \varepsilon$.*
*(b) If $|\mu_{\mathrm{L}} - \mu_{\mathrm{R}}|\sqrt{\eta(1-\eta)} > \sqrt{8\log(n/\varepsilon)/n}$, then $\mathbb{P}\{\|\mathcal{C}(\boldsymbol{X})\|_\infty \leq \sqrt{2\log(n/\varepsilon)}\} \leq \varepsilon$.*

For any $B > 0$, define

$$\Theta(B) := \left\{(\tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}}) \in [n-1] \times \mathbb{R} \times \mathbb{R} : |\mu_{\mathrm{L}} - \mu_{\mathrm{R}}|\sqrt{\tau(n-\tau)}/n \in \{0\} \cup (B, \infty)\right\}.$$

Here, $|\mu_{\mathrm{L}} - \mu_{\mathrm{R}}|\sqrt{\tau(n-\tau)}/n = |\mu_{\mathrm{L}} - \mu_{\mathrm{R}}|\sqrt{\eta(1-\eta)}$ can be interpreted as the signal-to-noise ratio of the mean change problem. Thus, $\Theta(B)$ is the parameter space of data distributions where there is either no change, or a single change-point in mean whose signal-to-noise ratio is at least $B$. The following corollary controls the misclassification risk of a CUSUM statistics-based classifier:

COROLLARY 4.1. *Fix $B > 0$. Let $\pi_0$ be any prior distribution on $\Theta(B)$, then draw $(\tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}}) \sim \pi_0$ and $\boldsymbol{X} \sim P(n, \tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}})$, and define $Y = \mathbb{1}\{\mu_{\mathrm{L}} \neq \mu_{\mathrm{R}}\}$. For $\lambda = B\sqrt{n}/2$, the classifier $h_\lambda^{\mathrm{CUSUM}}$ satisfies*

$$\mathbb{P}(h_\lambda^{\mathrm{CUSUM}}(\boldsymbol{X}) \neq Y) \leq ne^{-nB^2/8}.$$

Theorem 4.2 below, which is based on Corollary 4.1, Bartlett et al. (2019, Theorem 7) and Mohri et al. (2012, Corollary 3.4), shows that the empirical risk minimiser in the neural network class $\mathcal{H}_{1,2n-2}$ has good generalisation properties over the class of change-point problems parameterised by $\Theta(B)$. Given training data $(\boldsymbol{X}^{(1)}, Y^{(1)}), \ldots, (\boldsymbol{X}^{(N)}, Y^{(N)})$ and any $h : \mathbb{R}^n \to \{0, 1\}$, we define the empirical risk of $h$ as

$$L_N(h) := \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{Y^{(i)} \neq h(\boldsymbol{X}^{(i)})\}.$$

THEOREM 4.2. *Fix $B > 0$ and let $\pi_0$ be any prior distribution on $\Theta(B)$. We draw $(\tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}}) \sim \pi_0$, $\boldsymbol{X} \sim P(n, \tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}})$, and set $Y = \mathbb{1}\{\mu_{\mathrm{L}} \neq \mu_{\mathrm{R}}\}$. Suppose that the training data $\mathcal{D} := \big((\boldsymbol{X}^{(1)}, Y^{(1)}), \ldots, (\boldsymbol{X}^{(N)}, Y^{(N)})\big)$ consist of independent copies of $(\boldsymbol{X}, Y)$ and $h_{\mathrm{ERM}} := \arg\min_{h \in \mathcal{H}_{1,2n-2}} L_N(h)$ is the empirical risk minimiser. There exists a universal constant $C > 0$ such that for any $\delta \in (0, 1)$, (3) holds with probability $1 - \delta$.*

$$\mathbb{P}(h_{\mathrm{ERM}}(\boldsymbol{X}) \neq Y \mid \mathcal{D}) \leq n e^{-nB^2/8} + C\sqrt{\frac{n^2 \log(n) \log(N) + \log(1/\delta)}{N}}. \tag{3}$$

The theoretical results derived for the neural network-based classifier, here and below, all rely on the fact that the training and test data are drawn from the same distribution. However, we observe that in practice, even when the training and test sets have different error distributions, neural network-based classifiers still provide accurate results on the test set; see our discussion of Figure 2 in Section 5 for more details. The misclassification error in (3) is bounded by two terms. The first term represents the misclassification error of CUSUM-based classifier, see Corollary 4.1, and the second term depends on the complexity of the neural network class measured in its VC dimension. Theorem 4.2 suggests that for training sample size $N \gg n^2 \log n$, a well-trained single-hidden-layer neural network with $2n - 2$ hidden nodes would have comparable performance to that of the CUSUM-based classifier. However, as we will see in Section 5, in practice, a much smaller training sample size $N$ is needed for the neural network to be competitive in the change-point detection task. This is because the $2n - 2$ hidden layer nodes in the neural network representation of $h_\lambda^{\mathrm{CUSUM}}$ encode the components of the CUSUM transformation $(\pm \boldsymbol{v}_t^\top \boldsymbol{x} : t \in [n-1])$, which are highly correlated.

By suitably pruning the hidden layer nodes, we can show that a single-hidden-layer neural network with $O(\log n)$ hidden nodes is able to represent a modified version of the CUSUM-based classifier with essentially the same misclassification error. More precisely, let $Q := \lfloor \log_2(n/2) \rfloor$ and write $T_0 := \{2^q : 0 \leq q \leq Q\} \cup \{n - 2^q : 0 \leq q \leq Q\}$. We can then define

$$h_{\lambda^*}^{\mathrm{CUSUM}_*}(\boldsymbol{X}) = \mathbb{1}\Big\{\max_{t \in T_0} |\boldsymbol{v}_t^\top \boldsymbol{X}| > \lambda^*\Big\}.$$

By the same argument as in Lemma 3.1, we can show that $h_{\lambda^*}^{\mathrm{CUSUM}_*} \in \mathcal{H}_{1,4\lfloor \log_2(n) \rfloor}$ for any $\lambda^* > 0$. The following Theorem shows that high classification accuracy can be achieved under a weaker training sample size condition compared to Theorem 4.2.

THEOREM 4.3. *Fix $B > 0$ and let the training data $\mathcal{D}$ be generated as in Theorem 4.2. Let $h_{\mathrm{ERM}} := \arg\min_{h \in \mathcal{H}_{L,m}} L_N(h)$ be the empirical risk minimiser for a neural network*

*with $L \geq 1$ layers and $\boldsymbol{m} = (m_1, \ldots, m_L)^\top$ hidden layer widths. If $m_1 \geq 4\lfloor \log_2(n) \rfloor$ and $m_r m_{r+1} = O(n \log n)$ for all $r \in [L-1]$, then there exists a universal constant $C > 0$ such that for any $\delta \in (0,1)$, (4) holds with probability $1-\delta$.*

$$\mathbb{P}(h_{\mathrm{ERM}}(\boldsymbol{X}) \neq Y \mid \mathcal{D}) \leq 2\lfloor \log_2(n) \rfloor e^{-nB^2/24} + C\sqrt{\frac{L^2 n \log^2(Ln) \log(N) + \log(1/\delta)}{N}}. \quad (4)$$

Theorem 4.3 generalises the single hidden layer neural network representation in Theorem 4.2 to multiple hidden layers. In practice, multiple hidden layers help keep the misclassification error rate low even when $N$ is small, see the numerical study in Section 5. Theorems 4.2 and 4.3 are examples of how to derive generalisation errors of a neural network-based classifier in the change-point detection task. The same workflow can be employed in other types of changes, provided that suitable representation results of likelihood-based tests in terms of neural networks (e.g. Lemma 3.2) can be obtained. In a general result of this type, the generalisation error of the neural network will again be bounded by a sum of the error of the likelihood-based classifier together with a term originating from the VC-dimension bound of the complexity of the neural network architecture.

We further remark that for simplicity of discussion, we have focused our attention on data models where the noise vector $\boldsymbol{\xi} = \boldsymbol{X} - \mathbb{E}\boldsymbol{X}$ has independent and identically distributed normal components. However, since CUSUM-based tests are available for temporally correlated or sub-Weibull data, with suitably adjusted test threshold values, the above theoretical results readily generalise to such settings. See Theorems S4 and S6 in the online supplementary material for more details.

## 5. Numerical study

We now investigate empirically our approach of learning a change-point detection method by training a neural network. Motivated by the results from the previous section we will fit a neural network with a single layer and consider how varying the number of hidden layers and the amount of training data affects performance. We will compare to a test based on the CUSUM statistic, both for scenarios where the noise is independent and Gaussian, and for scenarios where there is auto-correlation or heavy-tailed noise. The CUSUM test can be sensitive to the choice of threshold, particularly when we do not have independent Gaussian noise, so we tune its threshold based on training data.

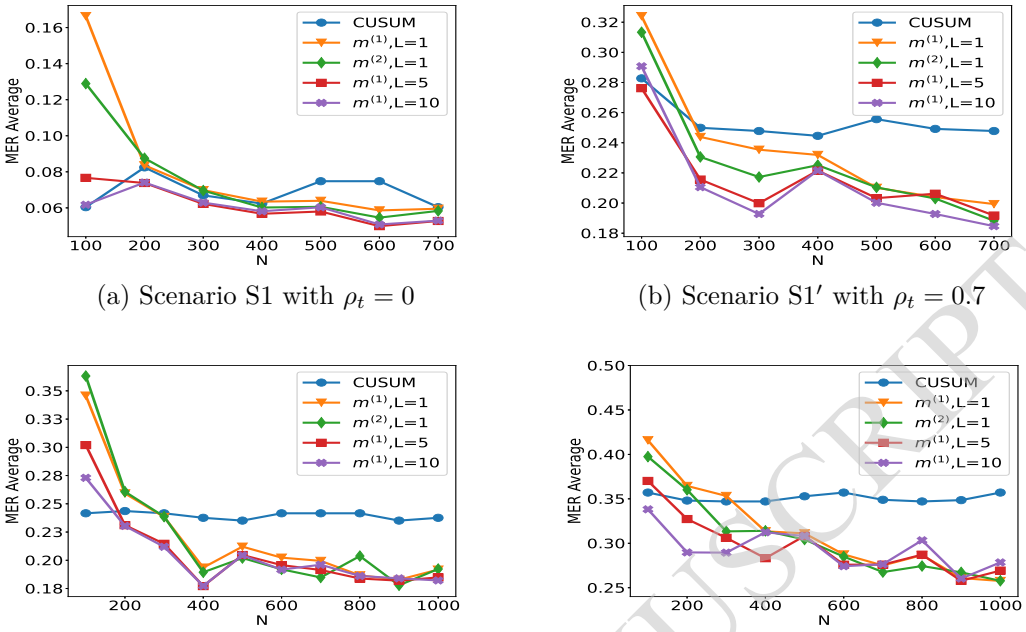
When training the neural network, we first standardise the data onto $[0,1]$, i.e. $\tilde{\boldsymbol{x}}_i = ((x_{ij} - x_i^{\min})/(x_i^{\max} - x_i^{\min}))_{j \in [n]}$ where $x_i^{\max} := \max_j x_{ij}, x_i^{\min} := \min_j x_{ij}$. This makes the neural network procedure invariant to either adding a constant to the data or scaling the data by a constant, which are natural properties to require. We train the neural network by minimising the cross-entropy loss on the training data. We run training for 200 epochs with a batch size of 32 and a learning rate of 0.001 using the Adam optimiser (Kingma and Ba, 2015). These hyperparameters are chosen based on a training dataset with cross-validation, more details can be found in Section 2 of the supplementary material.

We generate our data as follows. Given a sequence of length $n$, we draw $\tau \sim \mathrm{Unif}\{2, \ldots, n-2\}$, set $\mu_{\mathrm{L}} = 0$ and draw $\mu_{\mathrm{R}} | \tau \sim \mathrm{Unif}([-1.5b, -0.5b] \cup [0.5b, 1.5b])$, where $b := \sqrt{\frac{8n \log(20n)}{\tau(n-\tau)}}$

is chosen in line with Lemma 4.1 to ensure a good range of signal-to-noise ratios. We then generate $\boldsymbol{x}_1 = (\mu_{\mathrm{L}}\mathbb{1}_{\{t\le\tau\}} + \mu_{\mathrm{R}}\mathbb{1}_{\{t>\tau\}} + \varepsilon_t)_{t\in[n]}$, with the noise $(\varepsilon_t)_{t\in[n]}$ following an AR(1) model with possibly time-varying autocorrelation $\varepsilon_t|\rho_t = \xi_1$ for $t = 1$ and $\rho_t\varepsilon_{t-1} + \xi_t$ for $t \ge 2$, where $(\xi_t)_{t\in[n]}$ are independent, possibly heavy-tailed noise. The autocorrelations $\rho_t$ and innovations $\xi_t$ are from one of the three scenarios:

S1: $n = 100$, $N \in \{100, 200, \ldots, 700\}$, $\rho_t = 0$ and $\xi_t \sim N(0, 1)$.
S1′: $n = 100$, $N \in \{100, 200, \ldots, 700\}$, $\rho_t = 0.7$ and $\xi_t \sim N(0, 1)$.
S2: $n = 100$, $N \in \{100, 200, \ldots, 1000\}$, $\rho_t \sim \mathrm{Unif}([0, 1])$ and $\xi_t \sim N(0, 2)$.
S3: $n = 100$, $N \in \{100, 200, \ldots, 1000\}$, $\rho_t = 0$ and $\xi_t \sim \mathrm{Cauchy}(0, 0.3)$.

The above procedure is then repeated $N/2$ times to generate independent sequences $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N/2}$ with a single change, and the associated labels are $(y_1, \ldots, y_{N/2})^\top = \mathbf{1}_{N/2}$. We then repeat the process another $N/2$ times with $\mu_{\mathrm{R}} = \mu_{\mathrm{L}}$ to generate sequences without changes $\boldsymbol{x}_{N/2+1}, \ldots, \boldsymbol{x}_N$ with $(y_{N/2+1}, \ldots, y_N)^\top = \mathbf{0}_{N/2}$. The data with and without change $(\boldsymbol{x}_i, y_i)_{i\in[N]}$ are combined and randomly shuffled to form the training data. The test data are generated in a similar way, with a sample size $N_{\mathrm{test}} = 30000$ and the slight modification that $\mu_{\mathrm{R}}|\tau \sim \mathrm{Unif}([-1.75b, -0.25b] \cup [0.25b, 1.75b])$ when a change occurs. We note that the test data is drawn from the same distribution as the training set, though potentially having changes with signal-to-noise ratios outside the range covered by the training set. We have also conducted robustness studies to investigate the effect of training the neural networks on scenario S1 and test on S1′, S2 or S3. Qualitatively similar results to Figure 2 have been obtained in this misspecified setting (see Figure S2 of the online supplementary material). We compare the performance of the CUSUM-based classifier with the threshold cross-validated on the training data with neural networks from four function classes: $\mathcal{H}_{1,m^{(1)}}$,$\mathcal{H}_{1,m^{(2)}}$, $\mathcal{H}_{5,m^{(1)}\mathbf{1}_5}$ and $\mathcal{H}_{10,m^{(1)}\mathbf{1}_{10}}$ where $m^{(1)} = 4\lfloor\log_2(n)\rfloor$ and $m^{(2)} = 2n - 2$ respectively (cf. Theorem 4.3 and Lemma 3.1). Figure 2 shows the test misclassification error rate (MER) of the four procedures in the four scenarios S1, S1′, S2 and S3. We observe that when data are generated with independent Gaussian noise (Figure 2(a)), the trained neural networks with $m^{(1)}$ and $m^{(2)}$ single hidden layer nodes attain very similar test MER compared to the CUSUM-based classifier. This is in line with our Theorem 4.3. More interestingly, when noise has either autocorrelation (Figure 2(b, c)) or heavy-tailed distribution (Figure 2(d)), trained neural networks with $(L, \mathbf{m})$: $(1, m^{(1)})$, $(1, m^{(2)})$, $(5, m^{(1)}\mathbf{1}_5)$ and $(10, m^{(1)}\mathbf{1}_{10})$ outperform the CUSUM-based classifier, even after we have optimised the threshold choice of the latter. In addition, as shown in Figure S1 in the online supplement, when the first two layers of the network are set to carry out truncation, which can be seen as a composition of two ReLU operations, the resulting neural network outperforms the Wilcoxon statistics-based classifier (Dehling et al., 2015), which is a standard benchmark for change-point detection in the presence of heavy-tailed noise. Furthermore, from Figure 2, we see that increasing $L$ can significantly reduce the average MER when $N \le 200$. Theoretically, as the number of layers $L$ increases, the neural network is better able to approximate the optimal decision boundary, but it becomes increasingly difficult to train the weights due to issues such as vanishing gradients (He et al., 2016). A combination of these considerations leads us to develop deep neural network architecture with residual connections for detecting multiple changes and multiple change types in Section 6.




(a) Scenario S1 with $\rho_t = 0$

(b) Scenario S1′ with $\rho_t = 0.7$

(c) Scenario S2 with $\rho_t \sim \text{Unif}([0, 1])$

(d) Scenario S3 with Cauchy noise

**Fig. 2.** Plot of the test set MER, computed on a test set of size $N_{\text{test}} = 30000$, against training sample size $N$ for detecting the existence of a change-point on data series of length $n = 100$. We compare the performance of the CUSUM test and neural networks from four function classes: $\mathcal{H}_{1,m^{(1)}}$,$\mathcal{H}_{1,m^{(2)}}$, $\mathcal{H}_{5,m^{(1)}\mathbf{1}_5}$ and $\mathcal{H}_{10,m^{(1)}\mathbf{1}_{10}}$ where $m^{(1)} = 4\lfloor \log_2(n) \rfloor$ and $m^{(2)} = 2n - 2$ respectively under scenarios S1, S1′, S2 and S3 described in Section 5.

## 6. Detecting multiple changes and multiple change types – case study

From the previous section, we see that single and multiple hidden layer neural networks can represent CUSUM or generalised CUSUM tests and may perform better than likelihood-based test statistics when the model is misspecified. This prompted us to seek a general network architecture that can detect, and even classify, multiple types of change. Motivated by the similarities between signal processing and image recognition, we employed a deep convolutional neural network (CNN) (Yamashita et al., 2018) to learn the various features of multiple change-types. However, stacking more CNN layers cannot guarantee a better network because of vanishing gradients in training (He et al., 2016). Therefore, we adopted the residual block structure (He et al., 2016) for our neural network architecture. After experimenting with various architectures with different numbers of residual blocks and fully connected layers on synthetic data, we arrived at a network architecture with 21 residual blocks followed by a number of fully connected layers. Figure S5 of the online supplement shows an overview of the architecture of the final general-purpose deep neural network for change-point detection. The precise architecture and training methodology of this network $\widehat{NN}$ can be found in Section 3 of the supplement. Neural Architecture Search (NAS) approaches (see Paaß and Giesselbach, 2023, Section 2.4.3) offer principled

**Table 1.** Test classification accuracy of oracle likelihood-ratio based method ($\text{LR}^{\text{oracle}}$), adaptive likelihood ratio method ($\text{LR}^{\text{adapt}}$) and our residual neural network (NN) classifier for setups with weak and strong signal-to-noise ratios (SNR). Data are generated as a mixture of no change-point in mean or variance (Class 1), change in mean only (Class 2), change in variance only (Class 3), no-change in a non-zero slope (Class 4), change in slope only (Class 5). We report the true positive rate of each class and the accuracy in the last row.

| | Weak SNR | | | Strong SNR | | |
|---|---|---|---|---|---|---|
| | $\text{LR}^{\text{oracle}}$ | $\text{LR}^{\text{adapt}}$ | NN | $\text{LR}^{\text{oracle}}$ | $\text{LR}^{\text{adapt}}$ | NN |
| Class 1 | 0.9787 | 0.9457 | 0.8062 | 0.9787 | 0.9341 | 0.9651 |
| Class 2 | 0.8443 | 0.8164 | 0.8882 | 1.0000 | 0.7784 | 0.9860 |
| Class 3 | 0.8350 | 0.8291 | 0.8585 | 0.9902 | 0.9902 | 0.9705 |
| Class 4 | 0.9960 | 0.9453 | 0.8826 | 0.9980 | 0.9372 | 0.9312 |
| Class 5 | 0.8729 | 0.8604 | 0.8353 | 0.9958 | 0.9917 | 0.9147 |
| Accuracy | 0.9056 | 0.8796 | 0.8660 | 0.9924 | 0.9260 | 0.9672 |

ways of selecting neural architectures. Some of these approaches could be made applicable in our setting.

We demonstrate the power of our general purpose change-point detection network in a numerical study. We train the network on $N = 10000$ instances of data sequences generated from a mixture of no change-point in mean or variance, change in mean only, change in variance only, no-change in a non-zero slope and change in slope only, and compare its classification performance on a test set of size 2500 against that of oracle likelihood-based classifiers (where we pre-specify whether we are testing for change in mean, variance or slope) and adaptive likelihood-based classifiers (where we combine likelihood based tests using the Bayesian Information Criterion). Details of the data-generating mechanism and classifiers can be found in Section 2 of the supplementary material. The classification accuracy of the three approaches in weak and strong signal-to-noise ratio settings are reported in Table 1. We see that the neural network-based approach achieves similar classification accuracy as adaptive likelihood based method for weak SNR and higher classification accuracy than the adaptive likelihood based method for strong SNR. We would not expect the neural network to outperform the oracle likelihood-based classifiers as it has no knowledge of the exact change-type of each time series.

We now consider an application to detecting different types of change. The HASC (Human Activity Sensing Consortium) project data contain motion sensor measurements during a sequence of human activities, including "stay", "walk", "jog", "skip", "stair up" and "stair down". Complex changes in sensor signals occur during transition from one activity to the next (see Figure 3). We have 28 labels in HASC data, see Figure S6 of the supplement. To agree with the dimension of the output, we drop two dense layers "Dense(10)" and "Dense(20)" in Figure S5. The resulting network can be effectively applied for change-point detection in sensory signals of human activities, and can achieve high accuracy in change-point classification tasks (Figure S8 of supplementary material).

Finally, we remark that our neural network-based change-point detector can be utilised to detect multiple change-points. Algorithm 1 outlines a general scheme for turning a

change-point classifier into a location estimator, where we employ an idea similar to that of MOSUM (Eichinger and Kirch, 2018) and repeatedly apply a classifier $\psi$ to data from a sliding window of size $n$. Here, we require $\psi$ applied to each data segment $\boldsymbol{X}^*_{[i,i+n)}$ to output both the class label $L_i = 0$ or 1 if no change or a change is predicted and the corresponding probability $p_i$ of having a change. In our particular example, for each data segment $\boldsymbol{X}^*_{[i,i+n)}$ of length $n = 700$, we define $\psi(\boldsymbol{X}^*_{[i,i+n)}) = 0$ if $\widehat{NN}(\boldsymbol{X}^*_{[i,i+n)})$ predicts a class label in $\{0, 4, 8, 12, 16, 22\}$ (see Figure S6 in supplementary material) and 1 otherwise. The thresholding parameter $\gamma \in \mathbb{Z}^+$ is chosen to be $1/2$. Figure 4 illustrates the result of

**Algorithm 1:** Algorithm for change-point localisation

**Input:** new data $\boldsymbol{x}^*_1, \ldots, \boldsymbol{x}^*_{n^*} \in \mathbb{R}^d$, a trained classifier $\psi : \mathbb{R}^{d\times n} \to \{0, 1\}$, $\gamma > 0$.

1 Form $\boldsymbol{X}^*_{[i,i+n)} := (\boldsymbol{x}^*_i, \ldots, \boldsymbol{x}_{i+n-1})$ and compute $L_i \leftarrow \psi(\boldsymbol{X}^*_{[i,i+n)})$ for all $i = 1, \ldots, n^* - n + 1$;

2 Compute $\bar{L}_i \leftarrow n^{-1} \sum_{j=i-n+1}^{i} L_j$ for $i = n, \ldots, n^* - n + 1$;

3 Let $\{[s_1, e_1], \ldots, [s_{\hat{\nu}}, e_{\hat{\nu}}]\}$ be the set of all maximal segments such that $\bar{L}_i \geq \gamma$ for all $i \in [s_r, e_r]$, $r \in [\hat{\nu}]$ ;

4 Compute $\hat{\tau}_r \leftarrow \arg\max_{i\in[s_r,e_r]} \bar{L}_i$ for all $r \in [\hat{\nu}]$;

**Output:** Estimated change-points $\hat{\tau}_1, \ldots, \hat{\tau}_{\hat{\nu}}$

multiple change-point detection in HASC data which provides evidence that the trained neural network can detect both the multiple change-types and multiple change-points.

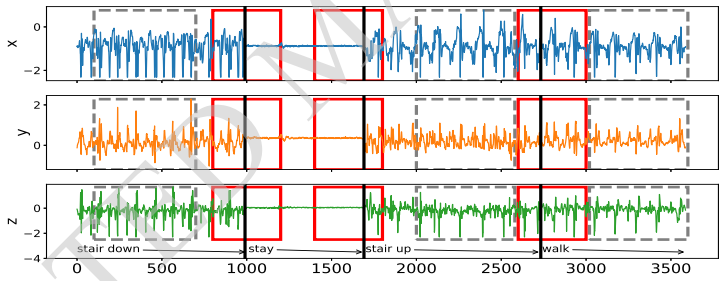



**Fig. 3.** The sequence of accelerometer data in $x, y$ and $z$ axes. From left to right, there are 4 activities: "stair down", "stay", "stair up" and "walk", their change-points are 990, 1691, 2733 respectively marked by black solid lines. The grey rectangles represent the group of "no-change" with labels: "stair down", "stair up" and "walk"; The red rectangles represent the group of "one-change" with labels: "stair down→stay", "stay→stair up" and "stair up→walk".

## 7. Discussion

Reliable testing for change-points and estimating their locations, especially in the presence of multiple change-points, other heterogeneities or untidy data, is typically a difficult problem for the applied statistician: they need to understand what type of change is sought, be able to characterise it mathematically, find a satisfactory stochastic model for

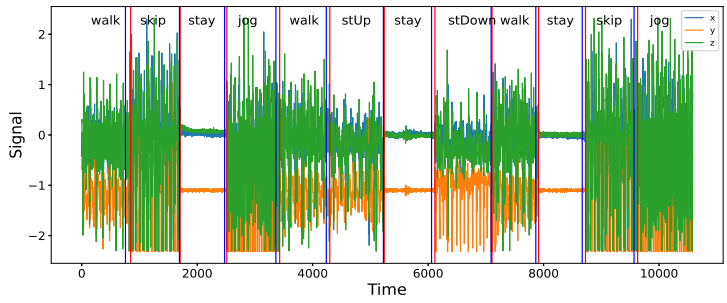



**Fig. 4.** Change-point detection in HASC data. The red vertical lines represent the underlying change-points, the blue vertical lines represent the estimated change-points. More details on multiple change-point detection can be found in Section 3 of supplementary material.

the data, formulate the appropriate statistic, and fine-tune its parameters. This makes for a long workflow, with scope for errors at its every stage.

In this paper, we showed how a carefully constructed statistical learning framework could automatically take over some of those tasks, and perform many of them 'in one go' when provided with examples of labelled data. This turned the change-point detection problem into a supervised learning problem, and meant that the task of learning the appropriate test statistic and fine-tuning its parameters was left to the 'machine' rather than the human user.

The crucial question was that of choosing an appropriate statistical learning framework. The key factor behind our choice of neural networks was the discovery that the traditionally-used likelihood-ratio-based change-point detection statistics could be viewed as simple neural networks, which (together with bounds on generalisation errors beyond the training set) enabled us to formulate and prove the corresponding learning theory. However, there are a plethora of other excellent predictive frameworks, such as XGBoost, LightGBM or Random Forests (Chen and Guestrin, 2016; Ke et al., 2017; Breiman, 2001) and it would be of interest to establish whether and why they could or could not provide a viable alternative to neural nets here. Furthermore, if we view the neural network as emulating the likelihood-ratio test statistic, in that it will create test statistics for each possible location of a change and then amalgamate these into a single classifier, then we know that test statistics for nearby changes will often be similar. This suggests that imposing some smoothness on the weights of the neural network may be beneficial.

A further challenge is to develop methods that can adapt easily to input data of different sizes, without having to train a different neural network for each input size. For changes in the structure of the mean of the data, it may be possible to use ideas from functional data analysis so that we pre-process the data, with some form of smoothing or imputation, to produce input data of the correct length.

If historical labelled examples of change-points, perhaps provided by subject-matter experts (who are not necessarily statisticians) are not available, one question of interest is whether simulation can be used to obtain such labelled examples artificially, based on (say) a single dataset of interest. Such simulated examples would need to come in two

flavours: one batch 'likely containing no change-points' and the other containing some artificially induced ones. How to simulate reliably in this way is an important problem, which this paper does not solve. Indeed, we can envisage situations in which simulating in this way may be easier than solving the original unsupervised change-point problem involving the single dataset at hand, with the bulk of the difficulty left to the 'machine' at the learning stage when provided with the simulated data.

For situations where there is no historical data, but there are statistical models, one can obtain training data by simulation from the model. In this case, training a neural network to detect a change has similarities with likelihood-free inference methods in that it replaces analytic calculations associated with a model by the ability to simulate from the model. It is of interest whether ideas from that area of statistics can be used here.

The main focus of our work was on testing for a single offline change-point, and we treated location estimation and extensions to multiple-change scenarios only superficially, via the heuristics of testing-based estimation in Section 6. Similar extensions can be made to the online setting once the neural network is trained, by retaining the final $n$ observations in an online stream in memory and applying our change-point classifier sequentially. One question of interest is whether and how these heuristics can be made more rigorous: equipped with an offline classifier only, how can we translate the theoretical guarantee of this offline classifier to that of the corresponding location estimator or online detection procedure? In addition to this approach, how else can a neural network, however complex, be trained to estimate locations or detect change-points sequentially? In our view, these questions merit further work.

## Availability of data and computer code

The data underlying this article are available in http://hasc.jp/hc2011/index-en.html. The computer code and algorithm are available in Python Package: AutoCPD.

## Acknowledgement

This work was supported by the High End Computing Cluster at Lancaster University, and EPSRC grants EP/V053590/1, EP/V053639/1 and EP/T02772X/1. We highly appreciate Yudong Chen's contribution to debug our Python scripts and improve their readability.

## Conflicts of Interest

We have no conflicts of interest to disclose.

## References

Ahmadzadeh, F. (2018). Change point detection with multivariate control charts by artificial neural network. *J. Adv. Manuf. Technol. 97*(9), 3179–3190.

Baranowski, R., Y. Chen, and P. Fryzlewicz (2019). Narrowest-over-threshold detection of multiple change points and change-point-like features. *J. Roy. Stat. Soc., Ser. B 81*(3), 649–672.

Bartlett, P. L., N. Harvey, C. Liaw, and A. Mehrabian (2019). Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *J. Mach. Learn. Res. 20*(63), 1–17.

Beaumont, M. A. (2019). Approximate Bayesian computation. *Annu. Rev. Stat. Appl. 6*, 379–403.

Bos, T. and J. Schmidt-Hieber (2022). Convergence rates of deep ReLU networks for multiclass classification. *Electron. J. Stat. 16*(1), 2724–2773.

Breiman, L. (2001). Random forests. *Mach. Learn. 45*(1), 5–32.

Chang, W.-C., C.-L. Li, Y. Yang, and B. Póczos (2019). Kernel change-point detection with auxiliary deep generative models. In *International Conference on Learning Representations*.

Chen, T. and C. Guestrin (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794.

De Ryck, T., M. De Vos, and A. Bertrand (2021). Change point detection in time series data using autoencoders with a time-invariant representation. *IEEE T. Signal Proces. 69*, 3513–3524.

Dehling, H., R. Fried, I. Garcia, and M. Wendler (2015). Change-point detection under dependence based on two-sample U-statistics. In D. Dawson, R. Kulik, M. Ould Haye, B. Szyszkowicz, and Y. Zhao (Eds.), *Asymptotic Laws and Methods in Stochastics: A Volume in Honour of Miklós Csörgő*, pp. 195–220. New York, NY: Springer New York.

Eichinger, B. and C. Kirch (2018). A MOSUM procedure for the estimation of multiple random change points. *Bernoulli 24*(1), 526–564.

Fearnhead, P., R. Maidstone, and A. Letchford (2019). Detecting changes in slope with an $l_0$ penalty. *J. Comput. Graph. Stat. 28*(2), 265–275.

Fearnhead, P. and G. Rigaill (2020). Relating and comparing methods for detecting changes in mean. *Stat 9*(1), 1–11.

Fryzlewicz, P. (2014). Wild binary segmentation for multiple change-point detection. *Ann. Stat. 42*(6), 2243–2281.

Fryzlewicz, P. (2021). Robust narrowest significance pursuit: Inference for multiple change-points in the median. *arXiv preprint*, arxiv:2109.02487.

Fryzlewicz, P. (2023). Narrowest significance pursuit: Inference for multiple change-points in linear models. *J. Am. Stat. Assoc.*, to appear.

Gao, Z., Z. Shang, P. Du, and J. L. Robertson (2019). Variance change point detection under a smoothly-changing mean trend with application to liver procurement. *J. Am. Stat. Assoc. 114*(526), 773–781.

Gourieroux, C., A. Monfort, and E. Renault (1993). Indirect inference. *J. Appl. Econom. 8*(S1), S85–S118.

Gupta, M., R. Wadhvani, and A. Rasool (2022). Real-time change-point detection: A deep neural network-based adaptive approach for detecting changes in multivariate time series data. *Expert Syst. Appl. 209*, 1–16.

Gutmann, M. U., R. Dutta, S. Kaski, and J. Corander (2018). Likelihood-free inference via classification. *Stat. Comput. 28*(2), 411–425.

Haynes, K., I. A. Eckley, and P. Fearnhead (2017). Computationally efficient changepoint detection for a range of penalties. *J. Comput. Graph. Stat. 26*(1), 134–143.

He, K., X. Zhang, S. Ren, and J. Sun (2016, June). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Hocking, T., G. Rigaill, and G. Bourque (2015). PeakSeg: constrained optimal segmentation and supervised penalty learning for peak detection in count data. In *International Conference on Machine Learning*, pp. 324–332. PMLR.

Huang, T.-J., Q.-L. Zhou, H.-J. Ye, and D.-C. Zhan (2023). Change point detection via synthetic signals. In *8th Workshop on Advanced Analytics and Learning on Temporal Data*.

James, B., K. L. James, and D. Siegmund (1987). Tests for a change-point. *Biometrika 74*(1), 71–83.

Jandhyala, V., S. Fotopoulos, I. MacNeill, and P. Liu (2013). Inference for single and multiple change-points in time series. *J. Time Ser. Anal. 34*(4), 423–446.

Ke, G., Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu (2017). LightGBM: A highly efficient gradient boosting decision tree. *Adv. Neur. In. 30*, 3146–3154.

Killick, R., P. Fearnhead, and I. A. Eckley (2012). Optimal detection of changepoints with a linear computational cost. *J. Am. Stat. Assoc. 107*(500), 1590–1598.

Kingma, D. P. and J. Ba (2015). Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun (Eds.), *ICLR (Poster)*.

Lee, J., Y. Xie, and X. Cheng (2023). Training neural networks for sequential change-point detection. In *IEEE ICASSP 2023*, pp. 1–5. IEEE.

Li, F., Z. Tian, Y. Xiao, and Z. Chen (2015). Variance change-point detection in panel data models. *Econ. Lett. 126*, 140–143.

Liehrmann, A., G. Rigaill, and T. D. Hocking (2021). Increased peak detection accuracy in over-dispersed ChIP-seq data with supervised segmentation models. *BMC Bioinform. 22*(1), 1–18.

Londschien, M., P. Bühlmann, and S. Kovács (2022). Random forests for change point detection. *arXiv preprint*, arxiv:2205.04997.

Mohri, M., A. Rostamizadeh, and A. Talwalkar (2012). *Foundations of Machine Learning*. Adaptive Computation and Machine Learning Series. Cambridge, MA: MIT Press.

Oh, K. J., M. S. Moon, and T. Y. Kim (2005). Variance change point detection via artificial neural networks for data separation. *Neurocomputing 68*, 239–250.

Paaß, G. and S. Giesselbach (2023). *Foundation Models for Natural Language Processing: Pre-trained Language Models Integrating Media*. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer International Publishing.

Picard, F., S. Robin, M. Lavielle, C. Vaisse, and J.-J. Daudin (2005). A statistical approach for array CGH data analysis. *BMC Bioinform. 6*(1).

Reeves, J., J. Chen, X. L. Wang, R. Lund, and Q. Q. Lu (2007). A review and comparison of changepoint detection techniques for climate data. *J. Appl. Meteorol. Clim. 46*(6), 900–915.

Ripley, B. D. (1994). Neural networks and related methods for classification. *J. Roy. Stat. Soc., Ser. B 56*(3), 409–456.

Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with ReLU activation function. *Ann. Stat. 48*(4), 1875–1897.

Shalev-Shwartz, S. and S. Ben-David (2014). *Understanding Machine Learning: From Theory to Algorithms*. New York, NY, USA: Cambridge University Press.

Truong, C., L. Oudre, and N. Vayatis (2020). Selective review of offline change point detection methods. *Signal Process. 167*, 107299.

Wang, T. and R. J. Samworth (2018). High dimensional change point estimation via sparse projection. *J. Roy. Stat. Soc., Ser. B 80*(1), 57–83.

Yamashita, R., M. Nishio, R. K. G. Do, and K. Togashi (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging 9*(4), 611–629.