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 detection methods based on training a neural network. Our approach is motivated by many existing tests for the presence of a change-point being able to be represented 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 test 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: Structural breaks; Neural networks; Likelihood-free inference; Supervised learning; Classification; Automatic statistician

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), and is seen as a key big-data problem (National Research Council, 2013).

Over the past few decades, various methodologies of change-point detection have been extensively studied, see Killick et al. (2012); Jandhyala et al. (2013); Fryzlewicz (2014); 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 for the type of change we wish to detect. This can lead to difficulties in practice. As we use new models for the data, 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 test learnt by such a neural network will give performance at least as good as 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 test 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). Within statistics, Londschien et al. (2022) and Lee et al. (2022) 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), but we are unaware of any previous work using such data to develop the change-point methods itself. As such, and for simplicity, we focus on the most fundamental aspect, namely the problem of detecting a single change. However, detecting and localising multiple changes is considered in Section 6 when analysing activity data.

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).

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 $\|\boldsymbol{x}\|_\infty := \max_i |x_i|$.

The rest of this article is organised as follows: Section 2 introduces some basic concepts of neural networks. We show that the CUSUM and other likelihood-ratio tests can be represented by a neural network in Section 3. We present theory about the generalisation error of neural network representations in Section 4. Section 5 shows the empirical performance of CUSUM and its neural network representations in different scenarios. In Section 6, we extend our methodology to multiple change-points and multiple change-types, and analyse human activity sensing data using a deep residual neural network. The paper ends with a discussion in Section 7. All proofs 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 direct 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 ith 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 $\sigma_{\boldsymbol{b}} : \mathbb{R}^r \to \mathbb{R}^r$:

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

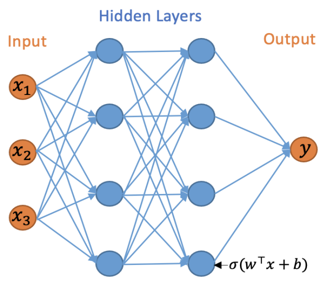


Fig. 1. A neural network with 2 hidden layers and width vector $\mathbf{m} = (4, 4)$

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

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

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 is helpful for understanding the Vapnik–Chervonenkis (VC) dimension (see, e.g. Shalev-Shwartz and Ben-David, 2014, Definition 6.5) of the neural network function class (Bartlett et al., 2019). In Theorem 4.3, the VC dimension of $\mathcal{H}_{L,\boldsymbol{m}}$ controls the generalisation error of the network under the assumption that we can obtain the true empirical risk minimiser. However, depending on the purpose, other output functions are possible such as the least-squares loss in nonparametric regression with deep neural networks (Schmidt-Hieber, 2020; Jiao et al., 2022) or truncated cross-entropy loss in multi-category classification (Bos and Schmidt-Hieber, 2022).

3. CUSUM test and its generalisations are neural networks

3.1. *Change in mean*

We initially consider the case of a single change-point $\tau \in [n-1]$ in the model

$$\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{\xi}, \tag{2}$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^\top := (\mu_\mathrm{L} \mathbb{1}\{i \le \tau\} + \mu_\mathrm{R} \mathbb{1}\{i > \tau\})_{i \in [n]} \in \mathbb{R}^n$ and $\mu_\mathrm{L}, \mu_\mathrm{R}$ are the signal 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 a 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 $\boldsymbol{v}_i := \Big(\sqrt{\frac{n-i}{in}} \mathbf{1}_i, -\sqrt{\frac{i}{(n-i)n}} \mathbf{1}_{n-i}\Big)^\top$ 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

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

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 test 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 CUSUM: other likelihood-ratio tests

We can generalise the simple change in mean model to allow for different types of change or for non-independent noise. Many such change-point models can be expressed as a change in regression problem, with the model for data given a change at τ being of the form

$$\boldsymbol{X} = \boldsymbol{Z}\boldsymbol{\beta} + \boldsymbol{c}_\tau \phi + \boldsymbol{\Gamma}\boldsymbol{\xi}, \tag{3}$$

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 τ, and the parameters $\boldsymbol{\beta}$ and ϕ 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 τ entries are zeros, and the remaining entries are ones. In this formulation β is the pre-change mean, and ϕ 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 ith entry is i; and $\boldsymbol{c}_\tau$ has ith entry that is $\max\{0, i - \tau\}$. In this formulation $\boldsymbol{\beta}$ defines the pre-change linear mean, and ϕ 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 (3) can be represented by our neural network.

Lemma 3.2. *Consider the change-point model (3) 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 test for change using a simple neural network.

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 an infinite 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 test, 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 $\boldsymbol{\mu} := (\mu_{\mathrm{L}}\mathbb{1}\{i \le \tau\} + \mu_{\mathrm{R}}\mathbb{1}\{i > \tau\})_{i\in[n]}$. Define $\eta := \tau/n$. Lemma 4.1 and Corollary 4.1 control the misclassification error of the CUSUM test.

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

For any $B > 0$, define

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

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 π_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 $Y = \mathbb{1}\{\mu_{\mathrm{L}} \ne \mu_{\mathrm{R}}\}$. For $\lambda = B\sqrt{n}/2$, the test $h_\lambda^{\mathrm{CUSUM}}$ satisfies*

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

Theorem 4.2 below, which is based on 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

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

We use the 0-1 empirical loss for ease of theoretical discussion as without the 0-1 loss, it is impossible to use the VC dimension to bound the generalisation error in Theorems 4.2 and 4.3. But we also remark that in practice we use the cross-entropy loss during training since its smoothness facilitates back-propagation updates of network weights.

Theorem 4.2. *Fix $B > 0$ and let π_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 $Y = \mathbb{1}\{\mu_{\mathrm{L}} \ne \mu_{\mathrm{R}}\}$. Suppose that the training data $(\boldsymbol{X}^{(1)}, Y^{(1)}), \ldots, (\boldsymbol{X}^{(N)}, Y^{(N)})$ are 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* $\delta \in (0,1)$, (4) *holds with probability* $1-\delta$.

$$\mathbb{P}(h_{\mathrm{ERM}}(\boldsymbol{X}) \neq Y) \leq ne^{-nB^2/8} + C\sqrt{\frac{n^2\log(n)\log(N)+\log(1/\delta)}{N}}. \tag{4}$$

The misclassification error in (4) is bounded by two terms. The first term represents the misclassification error of CUSUM test, 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 test. 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 test 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* $(\boldsymbol{X}^{(1)}, Y^{(1)}), \ldots, (\boldsymbol{X}^{(N)}, Y^{(N)})$ *be generated as in Theorem 4.2. Let* $h_{\mathrm{ERM}} := \arg\min_{h\in\mathcal{H}_{L,\boldsymbol{m}}} L_N(h)$ *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* $\delta \in (0,1)$, (5) *holds with probability* $1-\delta$.

$$\mathbb{P}(h_{\mathrm{ERM}}(\boldsymbol{X}) \neq Y) \leq 2\lfloor\log_2(n)\rfloor e^{-nB^2/24} + C\sqrt{\frac{L^2 n\log^2(Ln)\log(N)+\log(1/\delta)}{N}}. \tag{5}$$

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.

5. Numerical study

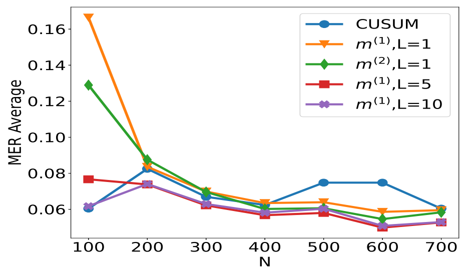
We now investigate empirically our approach of learning a change-point detection method by training a neural network. Motivated by the results from the previous section we will consider fitting a neural network with a single layer and consider how varying the number of hidden layers and the amount of 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, 2014). These hyperparameters are chosen based on a training dataset with cross-validation, more details can be found in Section 2 of the supplementary material.

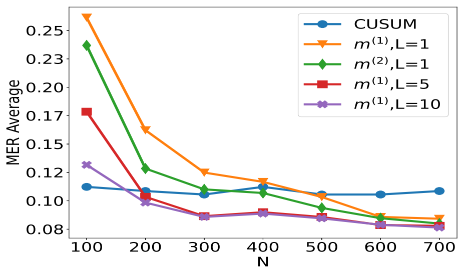
We generate our data as follows. Given a sequence of length n, we draw $\tau \sim \mathrm{Unif}\{2,\dots,n-2\}$, set $\mu_{\mathrm{L}} = 0$ and draw $\mu_{\mathrm{R}}|\tau \sim \mathrm{Unif}([-1.5b,-0.5b]\cup[0.5b,1.5b])$, where $b := \sqrt{\frac{8n\log(20n)}{\tau(n-\tau)}}$ is chosen in line with Lemma 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 ρ_t and innovations ξ_t are from one of the three scenarios:

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

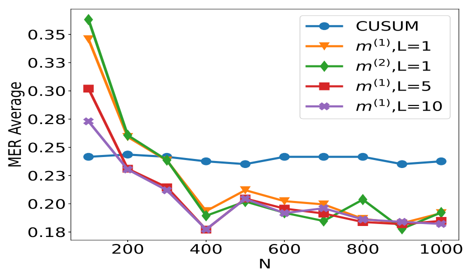
The above procedure is then repeated $N/2$ times to generate independent sequences $\boldsymbol{x}_1,\dots,\boldsymbol{x}_{N/2}$ with a single change, and the associated labels are $(y_1,\dots,y_{N/2})^\top = \mathbf{1}_{N/2}$. We then repeat the process another $N/2$ times with $\mu_{\mathrm{R}} = \mu_{\mathrm{L}}$ to generate sequences without changes $\boldsymbol{x}_{N/2+1},\dots,\boldsymbol{x}_N$ with $(y_{N/2+1},\dots,y_N)^\top = \mathbf{0}_{N/2}$. 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. This modification allows the test set to have changes with signal-to-noise ratios outside the range covered by the training set. We compare the performance of the CUSUM test 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 three scenarios S1, S2 and S3. We observe that when data are generated with independent Gaussian noise (Figure 2(a)), the trained neural network with $m^{(1)}$ and $m^{(2)}$ single hidden



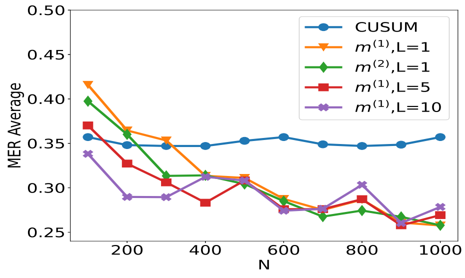
(a) Scenario S1 with $\rho_t = 0$



(b) Scenario S1 with $\rho_t = 0.7$



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



(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, S2 and S3 described in Section 5.

layer nodes attain very similar test MER compared to the CUSUM test, which is optimal in this setting. 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 test, even after we have optimised the threshold choice of the latter. Furthermore, given the width vector of neural network in Theorem 4.3, increasing L can significantly reduce the average MER when $N \leq 200$ (Figure 2). This leads us to develop more complex neural network architecture with finite training data for detecting multiple changes and multiple change types in Section 6.

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 3 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.

We demonstrate the power of our general purpose change-point detection network in a numerical study. We train the network on $N = 3000$ 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 30000 against that of oracle likelihood-based tests (where we pre-specify whether we are testing for change in mean, variance or slope) and adaptive likelihood-based tests (where we combine likelihood based tests using the Bayesian Information Criterion). Details of the data-generating mechanism and tests 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 higher classification accuracy than the adaptive likelihood based method. We would not expect the neural network to outperform the oracle likelihood-based tests as it has no knowledge of the exact change-type of each time series.

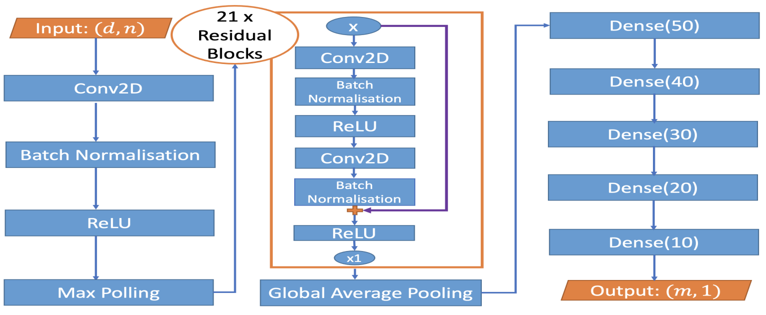


Fig. 3. Architecture of our general-purpose change-point detection neural network. The left column shows the standard layers of neural network with input size (d, n), d may represent the number of transformations or channels; We use 21 residual blocks and one global average pooling in the middle column; The right column includes 5 dense layers with nodes in bracket and output layer. More details of neural network architecture appear in supplementary material.

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 4). We have 28 labels in HASC data, see Section 3 of the supplement. To agree with the dimension of the output, we drop two dense layers

Table 1. Test classification accuracy of oracle likelihood-ratio based method (LR$^{\text{oracle}}$), adaptive likelihood ratio method (LR$^{\text{adapt}}$) and our residual neural network (ResNet) 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 recalls for each class and the accuracy in the last row.

	Weak SNR			Strong SNR		
	LR$^{\text{oracle}}$	LR$^{\text{adapt}}$	ResNet	LR$^{\text{oracle}}$	LR$^{\text{adapt}}$	ResNet
Class 1	0.9780	0.9548	0.8428	0.9749	0.9559	0.9551
Class 2	0.8278	0.8021	0.8012	1.0000	0.6496	0.9498
Class 3	0.7998	0.7973	1.0000	0.9789	0.9779	1.0000
Class 4	0.9953	0.9431	0.8813	0.9964	0.9431	0.9358
Class 5	0.8534	0.8362	0.8752	0.9841	0.9750	0.9548
Accuracy	0.8909	0.8667	0.8801	0.9869	0.9003	0.9591

"Dense(10)" and "Dense(20)" in Figure 3. The resulting network can be effectively applied for change-point detection in sensory signals of human activities, and can achieve high accuracy in single change-point classification tasks (Figure S5 of supplementary material).

Finally, we remark that our neural network-based change-point detector can be utilised to detect multiple change-points. We employ an idea similar to that of MOSUM (Eichinger and Kirch, 2018), where we apply the trained neural network to consecutive moving windows of bandwidth $n = 700$, which ensures there is at most one-change in each moving window. Then, we obtain a sequence of predicted change-types. The change-point location can be estimated from the endpoints of maximal contiguous windows in which the neural network classifies to yield no change. Algorithm 1 shows the pseudocode of change-point detection given the trained neural network $\widehat{NN}$ and unseen data $\mathcal{X}_{\text{new}}$ with size (d, n^*) where n^* is the length of sequential data and d may represent the number of transformations or channels. We apply Algorithm 1 with our trained network $\widehat{NN}$ on the HASC data. In this application, the output of $\widehat{NN}$ was relabelled to $\{0, 1\}$ to represent absence or presence of change detected, i.e. we relabel the pure activity labels such as "walk", "stay", etc as 0, meaning no-change and relabel transition labels such as "walk→stay", "Stair up→walk", etc as 1. Figure 5 illustrates the result of 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.

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 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.

Algorithm 1: Algorithm for change-point localisation

```
Input: new data $\mathcal{X}_{\text{new}} \in \mathbb{R}^{d\times n^*}$, a trained classifier $\widehat{NN} : \mathbb{R}^{d\times n} \to \{0,1\}$ and
       pre-specified $\gamma \in \mathbb{Z}^+$ to be the threshold of length of consecutive zeros. $\gamma$ is
       chosen based on the training dataset, more detail appears in Section 3 of the
       supplementary material.
1 for $i = 1, 2, \ldots, n^* - n + 1$ do
2     $\boldsymbol{x}^* \leftarrow \mathcal{X}_{\text{new}}[\,:\,,\ i : (i + n - 1)]$;
3     $L_i \leftarrow \widehat{NN}(\boldsymbol{x}^*)$;
4 end
5 Let $(s_1, e_1), \ldots, (s_\nu, e_\nu)$ be all pairs satisfying $e_j - s_j \geq \gamma$, $L_i = 0$ for all $i \in [s_j, e_j]$ and
    $L_{s_j-1} = L_{e_j+1} = 1$ (where $L_0 = L_{n^*-n+2} := 1$).;
6 $\tau_j^{\mathrm{L}} \leftarrow \min(e_j + n, s_{j+1}), \tau_j^{\mathrm{U}} \leftarrow \max(e_j + n, s_{j+1})$ for $j \in [\nu]$.;
Output: Interval estimators of change-points $\{[\tau_j^{\mathrm{L}}, \tau_j^{\mathrm{U}}] : j \in [\nu]\}$ and associated point
        estimators $\{\lfloor(\tau_j^{\mathrm{L}} + \tau_j^{\mathrm{U}})/2\rfloor : j \in \nu\}$
```

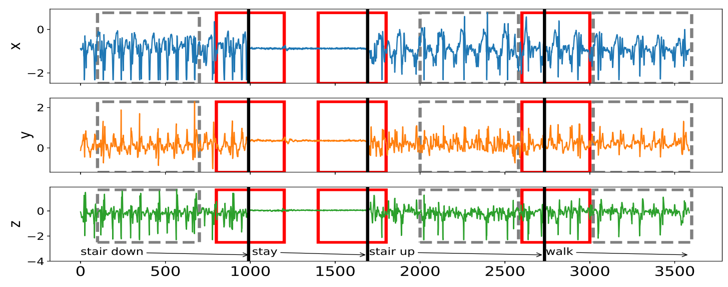


Fig. 4. The sequence of accelerometer data in x, y and z axes. From left to right, there are 4 activities: "stair down", "stay", "stair up" and "walk", their change-points are 990,1691,2733 respectively marked by black solid lines. The grey rectangles represent the group of "no-change" with labels: "stair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

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)

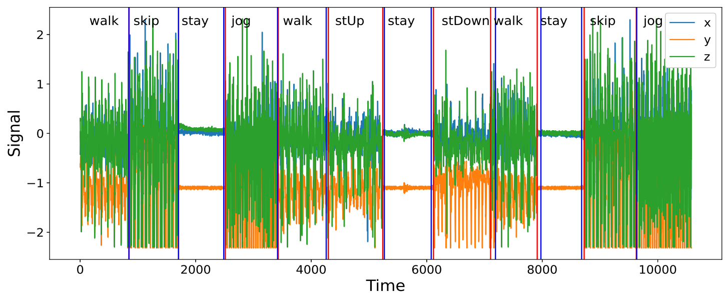


Fig. 5. Change-point detection of HASC data. The red vertical lines represent the underlying change-points, the blue vertical lines represent the estimated change-points. More results of multiple change-point detection can be found in Section 3 of supplementary material.

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 test, 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 change-points, and we treated location estimation only superficially, via the heuristics of testing-based estimation in Section 6. One question of interest here is whether and how these heuristics can be made more rigorous: equipped with a test only, in the unsupervised change-point detection context, this is the main idea behind the MOSUM approach (Eichinger and Kirch, 2018). In addition to this approach, how else can a neural network, however complex, be trained to estimate locations? In our view, these questions merit further work; in particular, we are interested in whether it is possible to employ the widely used one-hot encoding for the purpose of location estimation.

Finally, this paper focused on scenarios with a single change-point. While we offered heuristics for dealing with multiple change-points in our HASC case study of Section 6, this issue merits further study. Breaking up multiple change-point problems into single change-point problems is an established device in the change-point literature, see e.g. Baranowski et al. (2019); Anastasiou et al. (2022) and it would be of interest to examine their benefits in the supervised context.

Availability of data

The data underlying this article are available in http://hasc.jp/hc2011/index-en.html or GitHub repository: AutoCPD.

Acknowledgement

Jie Li acknowledges GPU resources provided by High End Computing Cluster at Lancaster University. This work is supported by EPSRC grants ref. nos. EP/V053590/1, EP/V053639/1 and EP/T02772X/1.

References

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

Anastasiou, A., I. Cribben, and P. Fryzlewicz (2022). Cross-covariance isolate detect: a new change-point method for estimating dynamic functional connectivity. *Medical Image Analysis 75*, 102252.

Baranowski, R., Y. Chen, and P. Fryzlewicz (2019). Narrowest-over-threshold detection of multiple change points and change-point-like features. *J. Roy. Statist. 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. *Annual Review of Statistics and its Application 6*, 379–403.

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

Breiman, L. (2001). Random forests. *Machine learning 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. *arXiv preprint*, arxiv:1901.06077.

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*, KDD '16, New York, NY, USA, pp. 785–794. Association for Computing Machinery.

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 Trans. Signal Process. 69*, 3513–3524.

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 l0 penalty. *J. Comput. Graph. Statist. 28*(2), 265–275.

Fearnhead, P. and G. Rigaill (2020). Relating and comparing methods for detecting changes in mean. *Stat 9*(1), e291.

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

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

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. Amer. Statist. Assoc. 114*(526), 773–781.

Gourieroux, C., A. Monfort, and E. Renault (1993). Indirect inference. *J. Appl. Econometrics 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 Systems with Applications 209*, 118260.

Gutmann, M. U., R. Dutta, S. Kaski, and J. Corander (2018). Likelihood-free inference via classification. *Statistics and Computing 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. Statist. 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.

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. *Journal of Time Series Analysis 34*(4), 423–446.

Jiao, Y., G. Shen, Y. Lin, and J. Huang (2022). Deep nonparametric regression on approximately low-dimensional manifolds. *arXiv preprint*, arxiv:2104.06708.

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. Inf. Process. Syst. 30*, 3146–3154.

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

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint*, arxiv:1412.6980.

Lee, J., Y. Xie, and X. Cheng (2022). Training neural networks for sequential change-point detection. *arXiv preprint*, arxiv:2210.17312.

Li, F., Z. Tian, Y. Xiao, and Z. Chen (2015). Variance change-point detection in panel data models. *Economics Letters 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 Bioinformatics 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.

National Research Council (2013). *Frontiers in Massive Data Analysis.* Washington, DC: The National Academies 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.

Picard, F., S. Robin, M. Lavielle, C. Vaisse, and J.-J. Daudin (2005). A statistical approach for array cgh data analysis. *BMC Bioinformatics 6*(1), 27.

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. *JAMC 46*(6), 900–915.

Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with relu activation function. *Ann. Statist. 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 Processing 167*, 107299.

Wang, T. and R. J. Samworth (2018). High dimensional change point estimation via sparse projection. *J. Roy. Statist. 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.

Supplementary Material: 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

This is the online supplementary material for the main paper Li, Fearnhead, Fryzlewicz, and Wang (2022), hereafter referred to as the main text. We present proofs of our main lemmas and theorems. Various technical details, results of numerical study and real data analysis are also listed here.

1. Proofs

1.1. *The proof of Lemma 3.1*

Define $W_0 := (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n-1}, -\boldsymbol{v}_1, \ldots, -\boldsymbol{v}_{n-1})^\top$ and $W_1 := \mathbf{1}_{2n-2}$, $\boldsymbol{b}_1 := \lambda \mathbf{1}_{2n-2}$ and $b_2 := 0$. Then $h(\boldsymbol{x}) := \sigma^*_{b_2} W_1 \sigma_{\boldsymbol{b}_1} W_0 \boldsymbol{x} \in \mathcal{H}_{1,2n-2}$ can be rewritten as

$$h(\boldsymbol{x}) = \mathbb{1}\left\{\sum_{i=1}^{n-1}\{(\boldsymbol{v}_i^\top \boldsymbol{x} - \lambda)_+ + (-\boldsymbol{v}_i^\top \boldsymbol{x} - \lambda)_+\} > b_2\right\} = \mathbb{1}\{\|\mathcal{C}(\boldsymbol{x})\|_\infty > \lambda\} = h_\lambda^{\mathrm{CUSUM}}(\boldsymbol{x}),$$

as desired.

1.2. *The Proof of Lemma 3.2*

As $\boldsymbol{\Gamma}$ is invertible, (3) in main text is equivalent to

$$\boldsymbol{\Gamma}^{-1}\boldsymbol{X} = \boldsymbol{\Gamma}^{-1}\boldsymbol{Z}\boldsymbol{\beta} + \boldsymbol{\Gamma}^{-1}\boldsymbol{c}_\tau \phi + \boldsymbol{\xi}.$$

Write $\tilde{\boldsymbol{X}} = \boldsymbol{\Gamma}^{-1}\boldsymbol{X}$, $\tilde{\boldsymbol{Z}} = \boldsymbol{\Gamma}^{-1}\boldsymbol{Z}$ and $\tilde{\boldsymbol{c}}_\tau = \boldsymbol{\Gamma}^{-1}\boldsymbol{c}_\tau$. If $\tilde{\boldsymbol{c}}_\tau$ lies in the column span of $\tilde{\boldsymbol{Z}}$, then the model with a change at τ is equivalent to the model with no change, and the likelihood-ratio test statistic will be 0. Otherwise we can assume, without loss of generality that $\tilde{\boldsymbol{c}}_\tau$ is orthogonal to each column of $\tilde{\boldsymbol{Z}}$: if this is not the case we can construct an equivalent model where we replace $\tilde{\boldsymbol{c}}_\tau$ with its projection to the space that is orthogonal to the column span of $\tilde{\boldsymbol{Z}}$.

As $\boldsymbol{\xi}$ is a vector of independent standard normal random variables, the likelihood-ratio statistic for a change at τ against no change is a monotone function of the reduction in the residual sum of squares of the model with a change at τ. The residual sum of squares of the no change model is

$$\tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{X}} - \tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{Z}}(\tilde{\boldsymbol{Z}}^\top \tilde{\boldsymbol{Z}})^{-1}\tilde{\boldsymbol{Z}}^\top \tilde{\boldsymbol{X}}.$$

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

The residual sum of squares for the model with a change at τ is

$$\tilde{\boldsymbol{X}}^\top\tilde{\boldsymbol{X}}-\tilde{\boldsymbol{X}}^\top[\tilde{\boldsymbol{Z}},\tilde{\boldsymbol{c}}_\tau]([\tilde{\boldsymbol{Z}},\tilde{\boldsymbol{c}}_\tau]^\top[\tilde{\boldsymbol{Z}},\tilde{\boldsymbol{c}}_\tau])^{-1}[\tilde{\boldsymbol{Z}},\tilde{\boldsymbol{c}}_\tau]^\top\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{X}}^\top\tilde{\boldsymbol{X}}-\tilde{\boldsymbol{X}}^\top\tilde{\boldsymbol{Z}}(\tilde{\boldsymbol{Z}}^\top\tilde{\boldsymbol{Z}})^{-1}\tilde{\boldsymbol{Z}}^\top\tilde{\boldsymbol{X}}-\tilde{\boldsymbol{X}}^\top\tilde{\boldsymbol{c}}_\tau(\tilde{\boldsymbol{c}}_\tau^\top\tilde{\boldsymbol{c}}_\tau)^{-1}\tilde{\boldsymbol{c}}_\tau^\top\tilde{\boldsymbol{X}}.$$

Thus, the reduction in residual sum of square of the model with the change at τ over the no change model is

$$\tilde{\boldsymbol{X}}^\top\tilde{\boldsymbol{c}}_\tau(\tilde{\boldsymbol{c}}_\tau^\top\tilde{\boldsymbol{c}}_\tau)^{-1}\tilde{\boldsymbol{c}}_\tau^\top\tilde{\boldsymbol{X}}=\left(\frac{1}{\sqrt{\tilde{\boldsymbol{c}}_\tau^\top\tilde{\boldsymbol{c}}_\tau}}\tilde{\boldsymbol{c}}_\tau^\top\tilde{\boldsymbol{X}}\right)^2$$

Thus if we define

$$\boldsymbol{v}_\tau=\frac{1}{\sqrt{\tilde{\boldsymbol{c}}_\tau^\top\tilde{\boldsymbol{c}}_\tau}}\tilde{\boldsymbol{c}}_\tau\boldsymbol{\Gamma}^{-1,}$$

then the likelihood-ratio test statistic is a monotone function of $|\boldsymbol{v}_\tau\boldsymbol{X}|$. This is true for all τ so the likelihood-ratio test is equivalent to

$$\max_{\tau\in[n-1]}|\boldsymbol{v}_\tau\boldsymbol{X}|>\lambda,$$

for some λ. This is of a similar form to the standard CUSUM test, except that the form of $\boldsymbol{v}_\tau$ is different. Thus, by the same argument as for Lemma 3.1 in main text, we can replicate this test with $h(\boldsymbol{x})\in\mathcal{H}_{1,2n-2}$, but with different weights to represent the different form for $\boldsymbol{v}_\tau$.

1.3. The Proof of Lemma 4.1

PROOF. (a) For each $i\in[n-1]$, since $\|\boldsymbol{v}_i\|_2=1$, we have $\boldsymbol{v}_i^\top\boldsymbol{X}\sim N(0,1)$. Hence, by the Gaussian tail bound and a union bound,

$$\mathbb{P}\Big\{\|\mathcal{C}(\boldsymbol{X})\|_\infty>t\Big\}\le\sum_{i=1}^{n-1}\mathbb{P}\left(\left|\boldsymbol{v}_i^\top\boldsymbol{X}\right|>t\right)\le n\exp(-t^2/2).$$

The result follows by taking $t=\sqrt{2\log(n/\varepsilon)}$.

(b) We write $\boldsymbol{X}=\boldsymbol{\mu}+\boldsymbol{Z}$, where $\boldsymbol{Z}\sim N_n(0,I_n)$. Since the CUSUM transformation is linear, we have $\mathcal{C}(\boldsymbol{X})=\mathcal{C}(\boldsymbol{\mu})+\mathcal{C}(\boldsymbol{Z})$. By part (a) there is an event Ω with probability at least $1-\varepsilon$ on which $\|\mathcal{C}(\boldsymbol{Z})\|_\infty\le\sqrt{2\log(n/\varepsilon)}$. Moreover, we have $\|\mathcal{C}(\boldsymbol{\mu})\|_\infty=|\boldsymbol{v}_\tau^\top\boldsymbol{\mu}|=|\mu_{\mathrm{L}}-\mu_{\mathrm{R}}|\sqrt{n\eta(1-\eta)}$. Hence on Ω, we have by the triangle inequality that

$$\|\mathcal{C}(\boldsymbol{X})\|_\infty\ge\|\mathcal{C}(\boldsymbol{\mu})\|_\infty-\|\mathcal{C}(\boldsymbol{Z})\|_\infty\ge|\mu_{\mathrm{L}}-\mu_{\mathrm{R}}|\sqrt{n\eta(1-\eta)}-\sqrt{2\log(n/\varepsilon)}>\sqrt{2\log(n/\varepsilon)},$$

as desired.

1.4. The Proof of Corollary 4.1

PROOF. From Lemma 4.1 in main text with $\varepsilon=ne^{-nB^2/8}$, we have

$$\mathbb{P}(h_\lambda^{\mathrm{CUSUM}}(\boldsymbol{X})\neq Y\mid\tau,\mu_{\mathrm{L}},\mu_{\mathrm{R}})\le ne^{-nB^2/8},$$

and the desired result follows by integrating over π_0.

1.5. Auxiliary Lemma

LEMMA 1.1. *Define* $T':=\{t_0\in\mathbb{Z}^+:|t_0-\tau|\le\min(\tau,n-\tau)/2\}$, *for any* $t_0\in T'$, *we have*

$$\min_{t_0\in T'}|\boldsymbol{v}_{t_0}^\top\boldsymbol{\mu}|\ge\frac{\sqrt{3}}{3}|\mu_{\mathrm{L}}-\mu_{\mathrm{R}}|\sqrt{n\eta(1-\eta)}.$$

PROOF. For simplicity, let $\Delta := |\mu_{\mathrm{L}} - \mu_{\mathrm{R}}|$, we can compute the CUSUM test statistics $a_i = |\boldsymbol{v}_i^\top \boldsymbol{\mu}|$ as:

$$a_i = \begin{cases} \Delta\,(1-\eta)\sqrt{\frac{ni}{n-i}} & 1 \le i \le \tau \\ \Delta\eta\sqrt{\frac{n(n-i)}{i}} & \tau < i \le n-1 \end{cases} \tag{1}$$

It is easy to verified that $a_\tau := \max_i(a_i) = \Delta\sqrt{n\eta(1-\eta)}$ when $i = \tau$. Next, we only discuss the case of $1 \le \tau \le \lfloor n/2 \rfloor$ as one can obtain the same result when $\lceil n/2 \rceil \le \tau \le n$ by the similar discussion.

When $1 \le \tau \le \lfloor n/2 \rfloor$, $|t_0 - \tau| \le \min(\tau, n-\tau)/2$ implies that $t_l \le t_0 \le t_u$ where $t_l := \lceil \tau/2 \rceil, t_u := \lfloor 3\tau/2 \rfloor$. Because a_i is an increasing function of i on $[1, \tau]$ and a decreasing function of i on $[\tau+1, n-1]$ respectively, the minimum of $a_{t_0}, t_l \le t_0 \le t_u$ happens at either t_l or t_u. Hence, we have

$$a_{t_l} \ge a_{\tau/2} = a_\tau\sqrt{\frac{n-\tau}{2n-\tau}}$$
$$a_{t_u} \ge a_{3\tau/2} = a_\tau\sqrt{\frac{2n-3\tau}{3(n-\tau)}}$$

Define $f(x) := \sqrt{\frac{n-x}{2n-x}}$ and $g(x) := \sqrt{\frac{2n-3x}{3(n-x)}}$. We notice that $f(x)$ and $g(x)$ are both decreasing functions of $x \in [1, n]$, therefore $f(\lfloor n/2 \rfloor) \ge f(n/2) = \sqrt{3}/3$ and $g(\lfloor n/2 \rfloor) \ge g(n/2) = \sqrt{3}/3$ as desired.

1.6. The Proof of Theorem 4.2

PROOF. Given any $L \ge 1$ and $\boldsymbol{m} = (m_1, \ldots, m_L)^\top$, let $m_0 := n$ and $m_{L+1} := 1$ and set $W^* = \sum_{r=1}^{L+1} m_{r-1}m_r$. Let $d := \mathrm{VCdim}(\mathcal{H}_{L,\boldsymbol{m}})$, then by Bartlett et al. (2019, Theorem 7), we have $d = O(LW^*\log(W^*))$. Thus, by Mohri et al. (2012, Corollary 3.4), for some universal constant $C > 0$, we have with probability at least $1-\delta$ that

$$\mathbb{P}(h_{\mathrm{ERM}}(\boldsymbol{X}) \ne Y) \le \min_{h \in \mathcal{H}_{L,\boldsymbol{m}}} \mathbb{P}(h(\boldsymbol{X}) \ne Y) + \sqrt{\frac{8d\log(2eN/d) + 8\log(4/\delta)}{N}}. \tag{2}$$

Here, we have $L = 1$, $m = 2n-2$, $W^* = O(n^2)$, so $d = O(n^2\log(n))$. In addition, since $h_\lambda^{\mathrm{CUSUM}} \in \mathcal{H}_{1,2n-2}$, we have $\min_{h \in \mathcal{H}_{L,\boldsymbol{m}}} \le \mathbb{P}(h_\lambda^{\mathrm{CUSUM}}(\boldsymbol{X}) \ne Y) \le ne^{-nB^2/8}$. Substituting these bounds into (2) we arrive at the desired result.

1.7. The Proof of Theorem 4.3

The following lemma, gives the mis-classification for the generalised CUSUM test where we only test for changes on a grid of $O(\log n)$ values.

LEMMA 1.2. *Fix $\varepsilon \in (0,1)$ and suppose that $\boldsymbol{X} \sim P(n, \tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}})$ for some $\tau \in [n-1]$ and $\mu_{\mathrm{L}}, \mu_{\mathrm{R}} \in \mathbb{R}$.*

(a) If $\mu_{\mathrm{L}} = \mu_{\mathrm{R}}$, then

$$\mathbb{P}\Big\{\max_{t \in T_0} |\boldsymbol{v}_t^\top \boldsymbol{X}| > \sqrt{2\log(|T_0|/\varepsilon)}\Big\} \le \varepsilon.$$

(b) If $|\mu_{\mathrm{L}} - \mu_{\mathrm{R}}|\sqrt{\eta(1-\eta)} > \sqrt{24\log(|T_0|/\varepsilon)/n}$, then we have

$$\mathbb{P}\Big\{\max_{t \in T_0} |\boldsymbol{v}_t^\top \boldsymbol{X}| \le \sqrt{2\log(|T_0|/\varepsilon)}\Big\} \le \varepsilon.$$

PROOF. (a) For each $i \in [n-1]$, since $\|\boldsymbol{v}_i\|_2 = 1$, we have $\boldsymbol{v}_i^\top \boldsymbol{X} \sim N(0,1)$. Hence, by the Gaussian tail bound and a union bound,

$$\mathbb{P}\Big\{\max_{i \in T_0} |\boldsymbol{v}_i^\top \boldsymbol{X}| > t\Big\} \le \sum_{i \in T_0} \mathbb{P}\left(\left|\boldsymbol{v}_i^\top \boldsymbol{X}\right| > t\right) \le |T_0|\exp(-t^2/2).$$

The result follows by taking $t = \sqrt{2\log(|T_0|/\varepsilon)}$.

(b) There exists some $t_0 \in T_0$ such that $|t_0 - \tau| \le \min\{\tau, n-\tau\}/2$. By Lemma 1.1, we have

$$|\boldsymbol{v}_{t_0}^\top \mathbb{E}\boldsymbol{X}| \ge \frac{\sqrt{3}}{3}\|\mathcal{C}(\mathbb{E}\boldsymbol{X})\|_\infty \ge \frac{\sqrt{3}}{3}|\mu_{\mathrm{L}} - \mu_{\mathrm{R}}|\sqrt{n\eta(1-\eta)} \ge 2\sqrt{2\log(|T_0|/\varepsilon)}.$$

Consequently, by the triangle inequality and result from part (a), we have with probability at least $1-\varepsilon$ that

$$\max_{t\in T_0} |\boldsymbol{v}_t^\top \boldsymbol{X}| \ge |\boldsymbol{v}_{t_0}^\top \boldsymbol{X}| \ge |\boldsymbol{v}_{t_0}^\top \mathbb{E}\boldsymbol{X}| - |\boldsymbol{v}_{t_0}^\top(\boldsymbol{X} - \mathbb{E}\boldsymbol{X})| \ge \sqrt{2\log(|T_0|/\varepsilon)},$$

as desired.

Using the above lemma we have the following result.

Corollary 1.1. *Fix $B > 0$. Let π_0 be any prior distribution on $\Theta(B)$, then draw $(\tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}}) \sim \pi_0$, $\boldsymbol{X} \sim P(n, \tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}})$, and define $Y = \mathbb{1}\{\mu_{\mathrm{L}} \ne \mu_{\mathrm{R}}\}$. Then for $\lambda^* = B\sqrt{n/12}$, the test $h_{\lambda^*}^{\mathrm{CUSUM}_*}$ satisfies*

$$\mathbb{P}(h_{\lambda^*}^{\mathrm{CUSUM}_*}(\boldsymbol{X}) \ne Y) \le 2\lfloor \log_2(n)\rfloor e^{-nB^2/24}.$$

Proof. Setting $\varepsilon = |T_0|e^{-nB^2/24}$ in Lemma 1.2, we have for any $(\tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}}) \in \Theta(B)$ that

$$\mathbb{P}(h_{\lambda^*}^{\mathrm{CUSUM}_*}(\boldsymbol{X}) \ne \mathbb{1}\{\mu_{\mathrm{L}} \ne \mu_{\mathrm{R}}\}) \le |T_0|e^{-nB^2/24}.$$

The result then follows by integrating over π_0 and the fact that $|T_0| = 2\lfloor \log_2(n)\rfloor$.

Proof (Proof of Theorem 4.3). We follow the proof of Theorem 4.2 up to (2). From the conditions of the theorem, we have $W^* = O(Ln\log n)$. Moreover, we have $h_{\lambda^*}^{\mathrm{CUSUM}_*} \in \mathcal{H}_{1,4\lfloor \log_2(n)\rfloor} \subseteq \mathcal{H}_{L,\boldsymbol{m}}$. Thus,

$$\begin{aligned}\mathbb{P}(h_{\mathrm{ERM}}(\boldsymbol{X}) \ne Y) &\le \mathbb{P}(h_{\lambda^*}^{\mathrm{CUSUM}_*}(\boldsymbol{X}) \ne Y) + C\sqrt{\frac{L^2 n\log n\log(Ln)\log(N) + \log(1/\delta)}{N}}\\ &\le 2\lfloor \log_2(n)\rfloor e^{-nB^2/24} + C\sqrt{\frac{L^2 n\log^2(Ln)\log(N) + \log(1/\delta)}{N}}\end{aligned}$$

as desired.

2. Simulation for Multiple Change-types

In this section, we illustrate the numerical study for one-change-point but with multiple change-types: change in mean, change in slope and change in variance.

The data set with change/no-change in mean is generated from $P(n, \tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}})$. We employ the model of change in slope from Fearnhead et al. (2019), namely

$$x_t = f_t + \xi_t = \begin{cases} \phi_0 + \phi_1 t + \xi_t & \text{if } 1 \le t \le \tau \\ \phi_0 + (\phi_1 - \phi_2)\tau + \phi_2 t + \xi_t & \tau + 1 \le t \le n, \end{cases} \tag{3}$$

where ϕ_0, ϕ_1 and ϕ_2 are parameters that can guarantee the continuity of two pieces of linear function at time $t = \tau$. We use the following model to generate the data set with change in variance.

$$y_t = \begin{cases} \mu + \varepsilon_t & \varepsilon_t \sim N(0, \sigma_1^2), & \text{if } t \le \tau \\ \mu + \varepsilon_t & \varepsilon_t \sim N(0, \sigma_2^2), & \text{otherwise} \end{cases}$$

where σ_1^2, σ_2^2 are the variances of two Gaussian distributions. τ is the change-point in variance. When $\sigma_1^2 = \sigma_2^2$, there is no-change in model. The labels of no change-point, change in mean only, change in variance only, no-change in variance and change in slope only are 0, 1, 2, 3, 4 respectively. For each label, we randomly generate N_{sub} time series. In each replication of N_{sub}, we update these parameters: $\tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}}, \sigma_1, \sigma_2, \alpha_1, \phi_1, \phi_2$. To avoid the boundary effect, we randomly choose τ from the discrete uniform distribution $U(n' + 1, n - n')$ in each replication, where $1 \le n' < \lfloor n/2\rfloor, n' \in \mathbb{N}$. The other parameters are generated as follows:

Table S1. The parameters for weak and strong signal-to-noise ratio (SNR).

Chang in mean	μ_l	μ_u	μ_{dl}	μ_{du}
Weak SNR	-5	5	0.25	0.5
Strong SNR	-5	5	0.8	1.2
Chang in variance	σ_l	σ_u	σ_{dl}	σ_{du}
Weak SNR	0.3	0.7	0.12	0.25
Strong SNR	0.3	0.7	0.2	0.4
Change in slope	ϕ_l	ϕ_u	ϕ_{dl}	ϕ_{du}
Weak SNR	-0.025	0.025	0.005	0.012
Strong SNR	-0.025	0.025	0.01	0.03

- $\mu_{\mathrm{L}}, \mu_{\mathrm{R}} \sim U(\mu_l, \mu_u)$ and $\mu_{dl} \le |\mu_{\mathrm{L}} - \mu_{\mathrm{R}}| \le \mu_{du}$, where μ_l, μ_u are the lower and upper bounds of $\mu_{\mathrm{L}}, \mu_{\mathrm{R}}$. μ_{dl}, μ_{du} are the lower and upper bounds of $|\mu_{\mathrm{L}} - \mu_{\mathrm{R}}|$.
- $\sigma_1, \sigma_2 \sim U(\sigma_l, \sigma_u)$ and $\sigma_{dl} \le |\sigma_1 - \sigma_2| \le \sigma_{du}$, where σ_l, σ_u are the lower and upper bounds of σ_1, σ_2. σ_{dl}, σ_{du} are the lower and upper bounds of $|\sigma_1 - \sigma_2|$.
- $\phi_1, \phi_2 \sim U(\phi_l, \phi_u)$ and $\phi_{dl} \le |\phi_1 - \phi_2| \le \phi_{du}$, where ϕ_l, ϕ_u are the lower and upper bounds of ϕ_1, ϕ_2. ϕ_{dl}, ϕ_{du} are the lower and upper bounds of $|\phi_1 - \phi_2|$.

Besides, we let $\mu = 20$, $\phi_0 = 0$ and the noise follows normal distribution with mean 0. For flexibility, we let the noise variance of change in mean and slope be 0.49 and 0.25 respectively. Both Scenarios 1 and 2 use the neural network architecture displayed in Figure 3 of main text.

Benchmark. Aminikhanghahi and Cook (2017) reviewed the methodologies for change-point detection in different types. To be simple, we employ the Pruned Exact Linear Time (PELT) (Killick et al., 2012) and Narrowest-Over-Threshold (NOT) (Baranowski et al., 2019) algorithms to detect the change in mean, slope and variance respectively. These two algorithms are available in **R** packages: **not** and **changepoint**. The oracle likelihood based tests $\mathrm{LR}^{\mathrm{oracle}}$ means that we pre-specified whether we are testing for change in mean, variance or slope. For the construction of adaptive likelihood-ratio based test $\mathrm{LR}^{\mathrm{adapt}}$, we first separately apply PELT and NOT algorithms to each time series, then we can compute 3 values of Bayesian information criterion (BIC) for each change-type based on the results of change-point detection. Lastly, the corresponding label of minimum of BIC values is treated as the predicted label.

Scenario 1: Weak SNR. Let $n = 400$, $N_{sub} = 600$ and $n' = 40$. The data is generated by the parameters settings in Table S1. We use the model architecture in Figure 3 of main text to train the classifier. The learning rate is 0.001, the batch size is 32, filter size in convolution layer is 16, the kernel size is $(4, 30)$, the epoch size is 400. The transformations are $(x, x^2, \log(x^2), \tanh(x))$. We also use the inverse time decay technique to dynamically reduce the learning rate. The result which is displayed in Table 1 of main text shows that the test accuracy of $\mathrm{LR}^{\mathrm{oracle}}$, $\mathrm{LR}^{\mathrm{adapt}}$ and ResNet based on 30000 test data sets are 0.8909, 0.8667 and 0.8801 respectively.

Scenario 2: Strong SNR. The parameters for generating strong-signal data are listed in Table S1. The other hyperparameters are same as in Scenario 1. the test accuracy of $\mathrm{LR}^{\mathrm{oracle}}$, $\mathrm{LR}^{\mathrm{adapt}}$ and ResNet based on 30000 test data sets are 0.9869, 0.9003 and 0.9591 respectively.. We can see that the neural network-based approach achieves higher classification accuracy than the adaptive likelihood based method.

3. Real Data Analysis

The HASC (Human Activity Sensing Consortium) project aims at understanding the human activities based on the sensor data. This data includes 6 human activities: “stay”, “walk”, “jog”, “skip”, “stair up” and “stair down”. Each activity lasts at least 10 seconds, the sampling frequency is 100 Hz.

3.1. Data Cleaning

The HASC offers sequential data where there are multiple change-types and multiple change-points, see Figure 4 in main text. Hence, we can not directly feed them into our deep convolutional residual neural network. The training data fed into our neural network requires fixed length n and either one changs-point or no change-point existence in each time serie. Next, we describe how to obtain this kind of training data from HASC sequential data. In general, Let $\boldsymbol{x} = (x_1, x_2, \dots, x_d)^\top, d \geq 1$ be the d-channel vector. Define $\boldsymbol{X} := (\boldsymbol{x}_{t_1}, \boldsymbol{x}_{t_2}, \dots, \boldsymbol{x}_{t_{n^*}})$ as a realization of d-variate time series where $\boldsymbol{x}_{t_j}, j = 1, 2, \dots, n^*$ are the observations of $\boldsymbol{x}$ at n^* consecutive time stamps $t_1, t_2, \dots, t_{n^*}$. Let $\boldsymbol{X}_i, i = 1, 2, \dots, N^*$ represent the observation from the i-th subject. $\boldsymbol{\tau}_i := (\tau_{i,1}, \tau_{i,2}, \dots, \tau_{i,K})^\top, K \in \mathbb{Z}^+, \tau_{i,k} \in [2, n^* - 1], 1 \leq k \leq K$ with convention $\tau_{i,0} = 0$ and $\tau_{i,K+1} = n^*$ represents the change-points of the i-th observation which are well-labelled in the sequential data sets. Furthermore, define $n := \min_{i \in [N^*]} \min_{k \in [K+1]} (\tau_{i,k} - \tau_{i,k-1})$. In practice, we require that n is not too small, this can be achieved by controlling the sampling frequency in experiment, see HASC data. We randomly choose q sub-segments with length n from $\boldsymbol{X}_i$ like the gray dash rectangles in Figure 4 of main text. By the definition of n, there is at most one change-point in each sub-segment. Meanwhile, we assign the label to each sub-segment according to the type and existence of change-point. After that, we stack all the sub-segments to form a tensor $\mathcal{X}$ with dimensions of (N^*q, d, n). The label vector is denoted as $\mathcal{Y}$ with length N^*q. To guarantee that there is at most one change-point in each segment, we set the length of segment $n = 700$. Let $q = 15$, as the change-points are well labelled, it is easy to draw 15 segments without any change-point, i.e., the segments with labels: “stay”, “walk”, “jog”, “skip”, “stair up” and “stair down”. Next, we randomly draw 15 segments (the red rectangles in Figure 4 of main text) for each transition point.

3.2. Transformation

Section 3 in main text suggests that changes in the mean/signal may be captured by feeding the raw data directly. For other type of change, we recommend appropriate transformations before training the model depending on the interest of change-type. For instance, if we are interested in changes in the second order structure, we suggest using the square transformation; for change in auto-correlation with order p we could input the cross-products of data up to a p-lag. In multiple change-types, we allow applying several transformations to the data in data pre-processing step. The mixture of raw data and transformed data is treated as the training data.

We employ the square transformation here. All the segments are mapped onto scale $[-1, 1]$ after the transformation. The frequency of training labels are list in Figure S2. Finally, the shapes of training and test data sets are $(4875, 6, 700)$ and $(1035, 6, 700)$ respectively.

3.3. Network Architecture

We propose a general deep convolutional residual neural network architecture to identify the multiple change-types based on the residual block technique (He et al., 2016). There are two reasons to explain why we choose residual block as the skeleton frame.

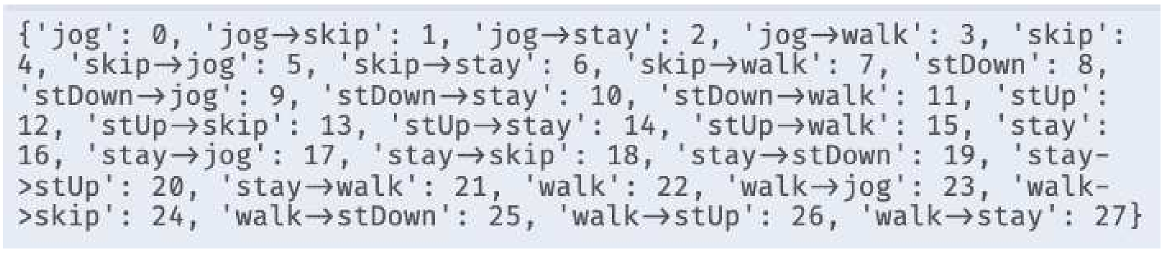
- The problem of vanishing gradients (Bengio et al., 1994; Glorot and Bengio, 2010). As the number of convolution layers goes significantly deep, some layer weights might vanish in back-propagation which hinders the convergence. Residual block can solve this issue by the so-called “shortcut connection”, see the flow chart in Figure 3 of main text.

- Degradation. He et al. (2016) has pointed out that when the number of convolution layers increases significantly, the accuracy might get saturated and degrade quickly. This phenomenon is reported and verified in He and Sun (2014) and He et al. (2016).

There are 21 residual blocks in our deep neural network, each residual block contains 2 convolutional layers. Like the suggestion in Ioffe and Szegedy (2015) and He et al. (2016), each convolution layer is followed by one Batch Normalization (BN) layer and one ReLU layer. Besides, there exist 5 fully-connected convolution layers right after the residual blocks, see the third column of Figure 3 in main text. For example, **Dense(50)** means that the dense layer has 50 nodes and is connected to a dropout layer with dropout rate 0.3. To further prevent the effect of overfitting, we also implement the L_2 regularization in each fully-connected layer (Ng, 2004). As the number of labels in HASC is 28, see Figure S1, we drop the dense layers "Dense(20)" and "Dense(10)" in Figure 3 of main text. The output layer has size $(28, 1)$.

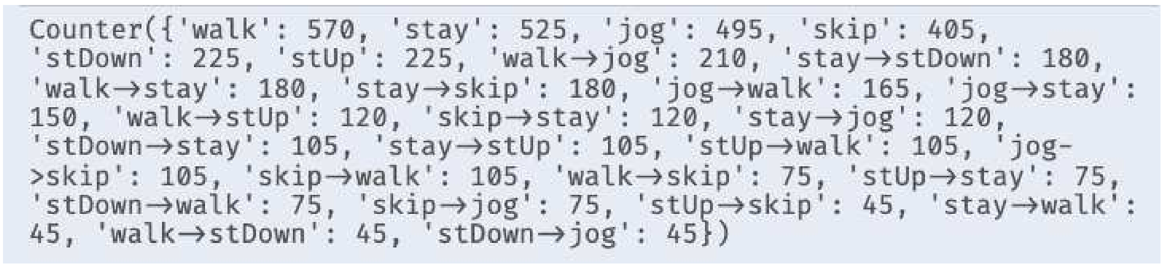
We remark two discussable issues here. (a) For other problems, the number of residual blocks, dense layers and the hyperparameters may vary depending on the complexity of the problem. In **??** of main text, the architecture of neural network for both synthetic data and real data has 21 residual blocks considering the trade-off between time complexity and model complexity. Like the suggestion in He et al. (2016), one can also add more residual blocks into the architecture to improve the accuracy of classification. (b) In practice, we would not have enough training data; but there would be potential ways to overcome this via either using Data Argumentation or increasing q. In some extreme cases that we only mainly have data with no-change, we can artificially add changes into such data in line with the type of change we want to detect.

3.4. *Training and Detection*



```
{'jog': 0, 'jog→skip': 1, 'jog→stay': 2, 'jog→walk': 3, 'skip':
4, 'skip→jog': 5, 'skip→stay': 6, 'skip→walk': 7, 'stDown': 8,
'stDown→jog': 9, 'stDown→stay': 10, 'stDown→walk': 11, 'stUp':
12, 'stUp→skip': 13, 'stUp→stay': 14, 'stUp→walk': 15, 'stay':
16, 'stay→jog': 17, 'stay→skip': 18, 'stay→stDown': 19, 'stay-
>stUp': 20, 'stay→walk': 21, 'walk': 22, 'walk→jog': 23, 'walk-
>skip': 24, 'walk→stDown': 25, 'walk→stUp': 26, 'walk→stay': 27}
```

Fig. S1. Label Dictionary



```
Counter({'walk': 570, 'stay': 525, 'jog': 495, 'skip': 405,
'stDown': 225, 'stUp': 225, 'walk→jog': 210, 'stay→stDown': 180,
'walk→stay': 180, 'stay→skip': 180, 'jog→walk': 165, 'jog→stay':
150, 'walk→stUp': 120, 'skip→stay': 120, 'stay→jog': 120,
'stDown→stay': 105, 'stay→stUp': 105, 'stUp→walk': 105, 'jog-
>skip': 105, 'skip→walk': 105, 'walk→skip': 75, 'stUp→stay': 75,
'stDown→walk': 75, 'skip→jog': 75, 'stUp→skip': 45, 'stay→walk':
45, 'walk→stDown': 45, 'stDown→jog': 45})
```

Fig. S2. Label Frequency

There are 7 persons observations in this dataset. The first 6 persons sequential data are treated as the training dataset, we use the last person's data to validate the trained classifier. Each person performs each of 6 activities: "stay", "walk", "jog", "skip", "stair up" and "stair down" at least 10 seconds. The transition point between two consecutive activities can be treated as the change-point. Therefore, there are 30 possible types of change-point. The total number of labels is 36 (6 activities and 30 possible transitions). However, we only found 28 different types of label in this real dataset, see Figure S1.

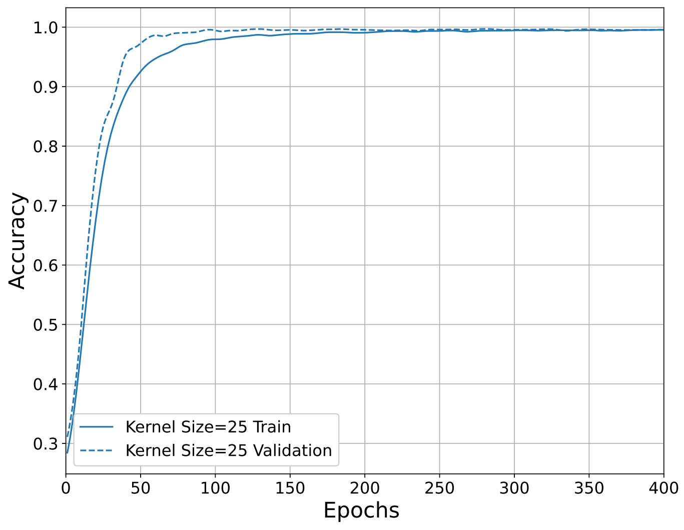


Fig. S3. The Accuracy Curves

The initial learning rate is 0.001, the epoch size is 400. Batch size is 16, the dropout rate is 0.3, the filter size is 16 and the kernel size is $(3, 25)$. Furthermore, we also use 20% of the training dataset to validate the classifier during training step.

Figure S3 shows the accuracy curves of training and validation. After 150 epochs, both solid and dash curves approximate to 1. The test accuracy is 0.9623, see the confusion matrix in Figure S4. These results show that our neural network classifier performs well both in the training and test datasets.

Next, we apply the trained classifier to 3 repeated sequential datasets of Person 7 to detect the change-points. The first sequential dataset has shape $(3, 10743)$. First, we extract the n-length sliding windows with stride 1 as the input dataset. The input size becomes $(9883, 6, 700)$. Second, we use Algorithm 1 to detect the change-points where we relabel the activity label as "no-change" label and transition label as "one-change" label respectively. The γ in Algorithm 1 is chosen by training data: Firstly, we applied the trained neural network to original sequential dataset of 6 persons; Secondly, choose γ using a grid search algorithm such that the mean absolute distance between true change-points and predicted change-points is minimal. Figures S5 and S6 show the results of multiple change-point detection for other 2 sequential data sets from the 7-th person.

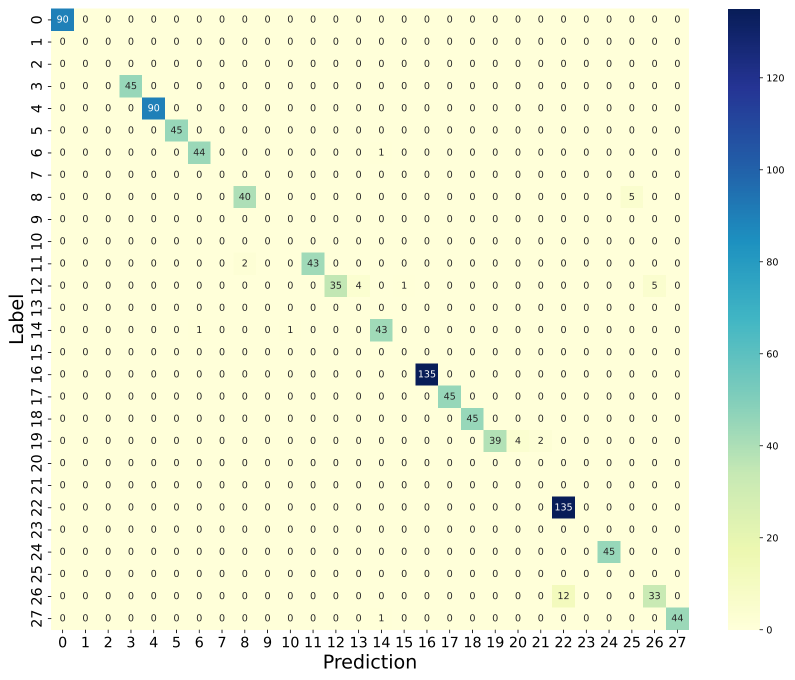


Fig. S4. Confusion Matrix of Real Test Dataset

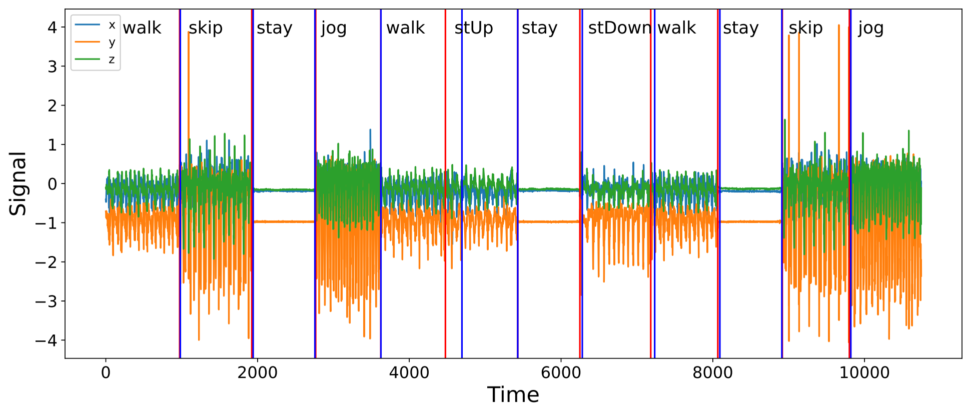


Fig. S5. Change-point Detection of Real Dataset for Person 7 (2nd sequence). The red line at 4476 is the true change-point, the blue line on its right is the estimator. The difference between them is caused by the similarity of “Walk” and “StairUp”.

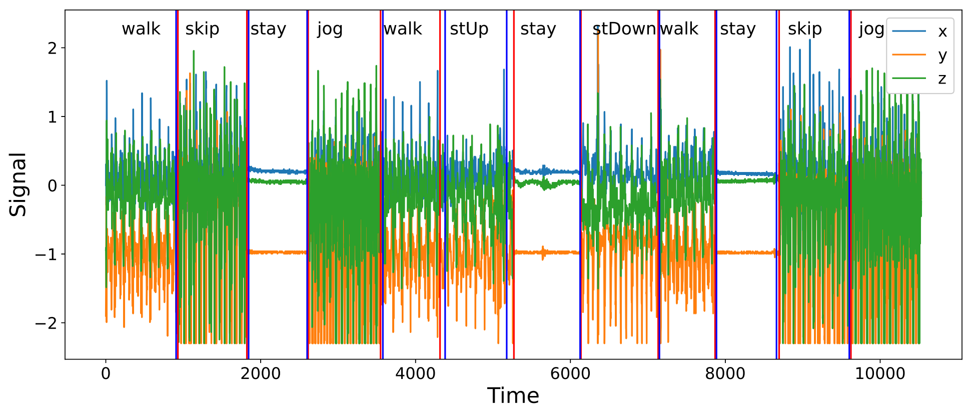


Fig. S6. Change-point Detection of Real Dataset for Person 7 (3rd sequence). The red vertical lines represent the underlying change-points, the blue vertical lines represent the estimated change-points.

References

Aminikhanghahi, S. and D. J. Cook (2017). Using change point detection to automate daily activity segmentation. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 262–267.

Baranowski, R., Y. Chen, and P. Fryzlewicz (2019). Narrowest-over-threshold detection of multiple change points and change-point-like features. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 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. *Journal of Machine Learning Research 20*(63), 1–17.

Bengio, Y., P. Simard, and P. Frasconi (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks 5*(2), 157–166.

Fearnhead, P., R. Maidstone, and A. Letchford (2019). Detecting changes in slope with an l0 penalty. *Journal of Computational and Graphical Statistics 28*(2), 265–275.

Glorot, X. and Y. Bengio (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings.

He, K. and J. Sun (2014). Convolutional neural networks at constrained time cost.

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

Ioffe, S. and C. Szegedy (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

Killick, R., P. Fearnhead, and I. A. Eckley (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association 107*(500), 1590–1598.

Li, J., P. Fearnhead, P. Fryzlewicz, and T. Wang (2022). Automatic change-point detection in time series via deep learning. *submitted*.

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

Ng, A. Y. (2004). Feature selection, l_1 vs. l_2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, New York, NY, USA, pp. 78. Association for Computing Machinery.