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

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

as desired.

1.2. The Proof of Lemma 3.2

As $\boldsymbol{\Gamma}$ is invertible, (2) 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

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

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

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

and the desired result follows by integrating over π_0.

1.5. Auxiliary Lemma

Lemma S1. *Define* $T'\coloneqq\{t_0\in\mathbb{Z}^+:|t_0-\tau|\le\min(\tau,n-\tau)/2\}$, *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 $\Delta \coloneqq |\mu_{\mathrm{L}} - \mu_{\mathrm{R}}|$, we can compute the CUSUM test statistics $a_i = |\boldsymbol{v}_i^\top \boldsymbol{\mu}|$ as:

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

It is easy to verified that $a_\tau \coloneqq \max_i(a_i) = \Delta\sqrt{n\eta(1-\eta)}$ 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 $t_l \coloneqq \lceil \tau/2 \rceil, t_u \coloneqq \lfloor 3\tau/2 \rfloor$. 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 $f(x) \coloneqq \sqrt{\frac{n-x}{2n-x}}$ and $g(x) \coloneqq \sqrt{\frac{2n-3x}{3(n-x)}}$. 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 $m_0 \coloneqq n$ and $m_{L+1} \coloneqq 1$ and set $W^* = \sum_{r=1}^{L+1} m_{r-1}m_r$. Let $d \coloneqq \mathrm{VCdim}(\mathcal{H}_{L,\boldsymbol{m}})$, 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

$$\mathbb{P}(h_{\mathrm{ERM}}(\boldsymbol{X}) \ne Y \mid \mathcal{D}) \le \min_{h \in \mathcal{H}_{L,\boldsymbol{m}}} \mathbb{P}(h(\boldsymbol{X}) \ne Y) + \sqrt{\frac{8d\log(2eN/d) + 8\log(4/\delta)}{N}}. \tag{S1}$$

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 (S1) we arrive at the desired result.

1.7. The Proof of Theorem 4.3

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

LEMMA S2. *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 $t \in [n-1]$, since $\|\boldsymbol{v}_t\|_2 = 1$, we have $\boldsymbol{v}_t^\top \boldsymbol{X} \sim N(0,1)$. Hence, by the Gaussian tail bound and a union bound,

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

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

(b) There exists some $t_0 \in T_0$ such that $|t_0 - \tau| \leq \min\{\tau, n-\tau\}/2$. By Lemma S1, we have

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

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

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

as desired.

Using the above lemma we have the following result.

COROLLARY S3. *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 $Y = \mathbb{1}\{\mu_{\mathrm{L}} \neq \mu_{\mathrm{R}}\}$. Then for $\lambda^* = B\sqrt{3n}/6$, the test $h_{\lambda^*}^{\mathrm{CUSUM}_*}$ satisfies*

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

PROOF. Setting $\varepsilon = |T_0|e^{-nB^2/24}$ in Lemma S2, we have for any $(\tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}}) \in \Theta(B)$ that

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

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 (S1). 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,

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

as desired.

1.8. Generalisation to time-dependent or heavy-tailed observations

So far, for simplicity of exposition, we have primarily focused on change-point models with independent and identically distributed Gaussian observations. However, neural network based procedures can also be applied to time-dependent or heavy-tailed observations. We first considered the case where the noise series $\xi_1, \ldots, \xi_n$ is a centred stationary Gaussian process with short-ranged temporal dependence. Specifically, writing $K(u) := \mathrm{cov}(\xi_t, \xi_{t+u})$, we assume that

$$\sum_{u=0}^{n-1} K(u) \leq D. \tag{S2}$$

THEOREM S4. *Fix $B > 0$, $n > 0$ and let π_0 be any prior distribution on $\Theta(B)$. We draw $(\tau, \mu_{\mathrm{L}}, \mu_{\mathrm{R}}) \sim \pi_0$, set $Y := \mathbb{1}\{\mu_{\mathrm{L}} \neq \mu_{\mathrm{R}}\}$ and generate $\boldsymbol{X} := \boldsymbol{\mu} + \boldsymbol{\xi}$ such that $\boldsymbol{\mu} := (\mu_{\mathrm{L}}\mathbb{1}\{i \leq \tau\} + \mu_{\mathrm{R}}\mathbb{1}\{i > \tau\})_{i\in[n]}$ and $\boldsymbol{\xi}$ is a centred stationary Gaussian process satisfying (S2). 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 let $h_{\mathrm{ERM}} := \arg\min_{h\in\mathcal{L}_{L,\boldsymbol{m}}} L_N(h)$ 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 for any $\delta \in (0,1)$, we have with probability at least $1-\delta$ that*

$$\mathbb{P}(h_{\mathrm{ERM}}(\boldsymbol{X}) \neq Y \mid \mathcal{D}) \leq 2\lfloor\log_2(n)\rfloor e^{-nB^2/(48D)} + C\sqrt{\frac{L^2 n\log^2(Ln)\log(N) + \log(1/\delta)}{N}}.$$

PROOF. By the proof of Wang and Samworth (2018, supplementary Lemma 10),

$$\mathbb{P}\big\{\max_{t\in T_0}|\boldsymbol{v}_t^\top\boldsymbol{\xi}| > B\sqrt{3n}/6\big\} \le |T_0|e^{-nB^2/(48D)}.$$

On the other hand, for t_0 defined in the proof of Lemma S1, we have that $|\mu_{\mathrm{L}}-\mu_{\mathrm{R}}|\sqrt{\tau(n-\tau)}/n > B$, then $|\boldsymbol{v}_{t_0}^\top\mathbb{E}X| \ge B\sqrt{3n}/3$. Hence for $\lambda^* = B\sqrt{3n}/6$, we have $h_{\lambda^*}^{\mathrm{CUSUM}_*}$ satisfying

$$\mathbb{P}(h_{\lambda^*}^{\mathrm{CUSUM}_*}(\boldsymbol{X}) \neq Y)) \le |T_0|e^{-nB^2/(48D)}.$$

We can then complete the proof using the same arguments as in the proof of Theorem 4.3.

We now turn to non-Gaussian distributions and recall that the Orlicz ψ_α-norm of a random variable Y is defined as

$$\|Y\|_{\psi_\alpha} := \inf\{\eta : \mathbb{E}\exp(|Y/\eta|^\alpha) \le 2\}.$$

For $\alpha \in (0,2)$, the random variable Y has heavier tail than a sub-Gaussian distribution. The following lemma is a direct consequence of Kuchibhotla and Chakrabortty (2022, Theorem 3.1) (We state the version used in Li et al. (2023, Proposition 14)).

LEMMA S5. *Fix $\alpha \in (0,2)$. Suppose $\boldsymbol{\xi} = (\xi_1,\dots,\xi_n)^\top$ has independent components satisfying $\mathbb{E}\xi_t = 0$, $\mathrm{Var}(\xi_t) = 1$ and $\|\xi_t\|_{\psi_\alpha} \le K$ for all $t \in [n]$. There exists $c_\alpha > 0$, depending only on α, such that for any $1 \le t \le n/2$, we have*

$$\mathbb{P}\big(|\boldsymbol{v}_t^\top\boldsymbol{\xi}| \ge y\big) \le \exp\left\{1 - c_\alpha \min\left\{\left(\frac{y}{K}\right)^2, \left(\frac{y}{K\|\boldsymbol{v}_t\|_{\beta(\alpha)}}\right)^\alpha\right\}\right\},$$

where $\beta(\alpha) = \infty$ for $\alpha \le 1$ and $\beta(\alpha) = \alpha/(\alpha-1)$ when $\alpha > 1$.

THEOREM S6. *Fix $\alpha \in (0,2)$, $B > 0$, $n > 0$ and let π_0 be any prior distribution on $\Theta(B)$. We draw $(\tau,\mu_{\mathrm{L}},\mu_{\mathrm{R}}) \sim \pi_0$, set $Y := \mathbb{1}\{\mu_{\mathrm{L}} \neq \mu_{\mathrm{R}}\}$ and generate $\boldsymbol{X} := \boldsymbol{\mu}+\boldsymbol{\xi}$ such that $\boldsymbol{\mu} := (\mu_{\mathrm{L}}\mathbb{1}\{i \le \tau\}+\mu_{\mathrm{R}}\mathbb{1}\{i > \tau\})_{i\in[n]}$ and $\boldsymbol{\xi} = (\xi_1,\dots,\xi_n)^\top$ satisfies $\mathbb{E}\xi_i = 0$, $\mathrm{Var}(\xi_i) = 1$ and $\|\xi_i\|_{\psi_\alpha} \le K$ for all $i \in [n]$. Suppose that the training data $\mathcal{D} := \big((\boldsymbol{X}^{(1)},Y^{(1)}),\dots,(\boldsymbol{X}^{(N)},Y^{(N)})\big)$ consist of independent copies of $(\boldsymbol{X},Y)$ and let $h_{\mathrm{ERM}} := \arg\min_{h\in\mathcal{L}_{L,\boldsymbol{m}}} L_N(h)$ be the empirical risk minimiser for a neural network with $L \ge 1$ layers and $\boldsymbol{m} = (m_1,\dots,m_L)^\top$ hidden layer widths. If $m_1 \ge 4\lfloor\log_2(n)\rfloor$ and $m_r m_{r+1} = O(n\log n)$ for all $r \in [L-1]$, then there exists a constant $c_\alpha > 0$, depending only on α such that for any $\delta \in (0,1)$, we have with probability at least $1-\delta$ that*

$$\mathbb{P}(h_{\mathrm{ERM}}(\boldsymbol{X}) \neq Y \mid \mathcal{D}) \le 2\lfloor\log_2(n)\rfloor e^{1-c_\alpha(\sqrt{n}B/K)^\alpha} + C\sqrt{\frac{L^2 n\log^2(Ln)\log(N)+\log(1/\delta)}{N}}.$$

PROOF. For $\alpha \in (0,2)$, we have $\beta(\alpha) > 2$, so $\|\boldsymbol{v}_t\|_{\beta(\alpha)} \ge \|\boldsymbol{v}_t\|_2 = 1$. Thus, from Lemma S5, we have $\mathbb{P}(|\boldsymbol{v}_t^\top\boldsymbol{\xi}| \ge y) \le e^{1-c_\alpha(y/K)^\alpha}$. Thus, following the proof of Corollary S3, we can obtain that $\mathbb{P}(h_{\lambda^*}^{\mathrm{CUSUM}_*}(\boldsymbol{X} \neq Y)) \le 2\lfloor\log_2(n)\rfloor e^{1-c_\alpha(\sqrt{n}B/K)^\alpha}$. Finally, the desired conclusion follows from the same argument as in the proof of Theorem 4.3.

1.9. Multiple change-point estimation

Algorithm 1 is a general scheme for turning a change-point classifier into a location estimator. While it is theoretically challenging to derive theoretical guarantees for the neural network based change-point location estimation error, we motivate this methodological proposal here by showing that Algorithm 1, applied in conjunction with a CUSUM-based classifier have optimal rate of convergence for the change-point localisation task. We consider the model $x_i = \mu_i + \xi_i$, where $\xi_i \overset{\mathrm{iid}}{\sim} N(0,1)$ for $i \in [n^*]$. Moreover, for a sequence of change-points $0 = \tau_0 < \tau_1 < \cdots < \tau_\nu < n = \tau_{\nu+1}$ satisfying $\tau_r - \tau_{r-1} \ge 2n$ for all $r \in [\nu+1]$ we have $\mu_i = \mu^{(r-1)}$ for all $i \in [\tau_{r-1},\tau_r]$, $r \in [\nu+1]$.

THEOREM S7. *Suppose data $x_1, \ldots, x_{n^*}$ are generated as above satisfying $|\mu^{(r)} - \mu^{(r-1)}| > 2\sqrt{2}B$ for all $r \in [\nu]$. Let $h_{\lambda^*}^{\mathrm{CUSUM}_*}$ be defined as in Corollary S3. Let $\hat{\tau}_1, \ldots, \hat{\tau}_{\hat{\nu}}$ be the output of Algorithm 1 with input $x_1, \ldots, x_{n^*}$, $\psi = h_{\lambda^*}^{\mathrm{CUSUM}_*}$ and $\gamma = \lfloor n/2 \rfloor / n$. Then we have*

$$\mathbb{P}\left\{\hat{\nu} = \nu \text{ and } |\tau_i - \hat{\tau}_i| \leq \frac{2B^2}{|\mu^{(r)} - \mu^{(r-1)}|^2}\right\} \geq 1 - 2n^* \lfloor \log_2(n) \rfloor e^{-nB^2/24}.$$

PROOF. For simplicity of presentation, we focus on the case where n is a multiple of 4, so $\gamma = 1/2$. Define

$$I_0 := \{i : \mu_{i+n-1} = \mu_i\},$$
$$I_1 := \left\{i : |\mu_{i+n-1} - \mu_i| \max_{r \in [\nu]} \sqrt{\frac{(\tau_r - i)(i + n - \tau_r)}{n^2}} \geq B\right\}.$$

By Lemma S2 and a union bound, the event

$$\Omega = \left\{h_{\lambda^*}^{\mathrm{CUSUM}_*}(\boldsymbol{X}^*_{[i,i+n)}) = k, \text{ for all } i \in I_k, \, k = 0, 1\right\}$$

has probability at least $1 - 2n^* \lfloor \log_2(n) \rfloor e^{-nB^2/24}$. We work on the event Ω henceforth. Denote $\Delta_r := 2B^2/|\mu^{(r)} - \mu^{(r-1)}|^2$. Since $|\mu^{(r)} - \mu^{(r-1)}| > 2\sqrt{2}B$, we have $\Delta_r < n/4$. Note that for each $r \in [\nu]$, we have $\{i : \tau_{r-1} < i \leq \tau_r - n \text{ or } \tau_r < i \leq \tau_{r+1} - n\} \subseteq I_0$ and $\{i : \tau_r - n + \Delta_r < i \leq \tau_r - \Delta_r\} \subseteq I_1$. Consequently, $\bar{L}_i$ defined in Algorithm 1 is below the threshold $\gamma = 1/2$ for all $i \in (\tau_{r-1} + n/2, \tau_r - n/2] \cup (\tau_r + n/2, \tau_{r+1} - n/2]$, monotonically increases for $i \in (\tau_r - n/2, \tau_r - \Delta]$ and monotonically decreases for $i \in (\tau_r + \Delta, \tau_r + n/2]$ and is above the threshold γ for $i \in (\tau_r - \Delta, \tau_r + \Delta]$. Thus, exactly one change-point, say $\hat{\tau}_r$, will be identified on $(\tau_{r-1} + n/2, \tau_{r+1} - n/2]$ and $\hat{\tau}_r = \arg\max_{i \in (\tau_{r-1} + n/2, \tau_{r+1} - n/2]} \bar{L}_i \in (\tau_r - \Delta, \tau_r + \Delta]$ as desired. Since the above holds for all $r \in [\nu]$, the proof is complete.

Assuming that $\log(n^*) \asymp \log(n)$ and choosing B to be of order $\sqrt{\log n}$, the above theorem shows that using the CUSUM-based change-point classifier $\psi = h_{\lambda^*}^{\mathrm{CUSUM}_*}$ in conjunction with Algorithm 1 allows for consistent estimation of both the number of locations of multiple change-points in the data stream. In fact, the rate of estimating each change-point, $2B^2/|\mu^{(r)} - \mu^{(r-1)}|^2$, is minimax optimal up to logarithmic factors (see, e.g. Verzelen et al., 2020, Proposition 6). An inspection of the proof of Theorem S7 reveals that the same result would hold for any ψ for which the event Ω holds with high probability. In view of the representability of $h_{\lambda^*}^{\mathrm{CUSUM}_*}$ in the class of neural networks, one would intuitively expect that a similar theoretical guarantee as in Theorem S7 would be available to the empirical risk minimiser in the corresponding neural network function class. However, the particular way in which we handle the generalisation error in the proof of Theorem 4.3 makes it difficult to proceed in this way, due to the fact that the distribution of the data segments obtained via sliding windows have complex dependence and no longer follow a common prior distribution π_0 used in Theorem 4.2.

2. Simulation and Result

2.1. 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

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

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.

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

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	0.6	1.2
Chang in variance	σ_l	σ_u	σ_{dl}	σ_{du}
Weak SNR	0.3	0.7	0.12	0.24
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	0.006	0.012
Strong SNR	-0.025	0.025	0.015	0.03

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 $1 \leq n' < \lfloor n/2 \rfloor, n' \in \mathbb{N}$. The other parameters are generated as follows:

- $\mu_{\mathrm{L}}, \mu_{\mathrm{R}} \sim U(\mu_l, \mu_u)$ and $\mu_{dl} \leq |\mu_{\mathrm{L}} - \mu_{\mathrm{R}}| \leq \mu_{du}$, 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 $\sigma_{dl} \leq |\sigma_1 - \sigma_2| \leq \sigma_{du}$, 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 $\phi_{dl} \leq |\phi_1 - \phi_2| \leq \phi_{du}$, 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 $\mu = 0$, $\phi_0 = 0$ 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 defined below use the neural network architecture displayed in Figure S5.

Benchmark. Aminikhanghahi and Cook (2017) reviewed the methodologies for change-point detection in different types. To be simple, we employ the Narrowest-Over-Threshold (NOT) (Baranowski et al., 2019) and single variance change-point detection (Chen and Gupta, 2012) 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 3 detection algorithms of change in mean, variance and slope 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 $n = 400$, $N_{sub} = 2000$ and $n' = 40$. The data is generated by the parameters settings in Table S1. We use the model architecture in Figure S5 to train the classifier. The learning rate is 0.001, the batch size is 64, filter size in convolution layer is 16, the kernel size is $(3, 30)$, the epoch size is 500. The transformations are (x, x^2). 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 2500 test data sets are 0.9056, 0.8796 and 0.8660 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 2500 test data sets are 0.9924, 0.9260 and 0.9672 respectively. We can see that the neural network-based approach achieves higher classification accuracy than the adaptive likelihood based method.

Table S2. Test classification accuracy of likelihood-ratio (LR) based classifier (Chen and Gupta, 2012, p.59) and our residual neural network (ResNet) based classifier with 21 residual blocks for setups with weak and strong signal-to-noise ratios (SNR). Data are generated as a mixture of no change-point (Class 1), change in mean and variance at a same change-point (Class 2). We report the true positive rate of each class and the accuracy in the last row. The optimal threshold value of LR is chosen by the grid search method on the training dataset.

	Weak SNR		Strong SNR	
	LR	ResNet	LR	ResNet
Class 1	0.9823	0.9668	1.0000	0.9991
Class 2	0.8759	0.9621	0.9995	0.9992
Accuracy	0.9291	0.9645	0.9997	0.9991

2.2. Some Additional Simulations

2.2.1. Simulation for simultaneous changes

In this simulation, we compare the classification accuracies of likelihood-based classifier and ResNet-based classifier under the circumstance of simultaneous changes. For simplicity, we only focus on two classes: no change-point (Class 1) and change in mean and variance at a same change-point (Class 2). The change-point location τ is randomly drawn from Unif$\{40, \ldots, n-41\}$ where $n = 400$ is the length of time series. Given τ, to generate the data of Class 2, we use the parameter settings of change in mean and change in variance in Table S1 to randomly draw $\mu_{\mathrm{L}}, \mu_{\mathrm{R}}$ and σ_1, σ_2 respectively. The data before and after the change-point τ are generated from $N(\mu_{\mathrm{L}}, \sigma_1^2)$ and $N(\mu_{\mathrm{R}}, \sigma_2^2)$ respectively. To generate the data of Class 1, we just draw the data from $N(\mu_{\mathrm{L}}, \sigma_1^2)$. Then, we repeat each data generation of Class 1 and 2 2500 times as the training dataset. The test dataset is generated in the same procedure as the training dataset, but the testing size is 15000. We use two classifiers: likelihood-ratio (LR) based classifier (Chen and Gupta, 2012, p.59) and a 21-residual-block neural network (ResNet) based classifier displayed in Figure S5 to evaluate the classification accuracy of simultaneous change v.s. no change. The result are displayed in Table S2. We can see that under weak SNR, the ResNet has a good performance than LR-based method while it performs as well as the LR-based method under strong SNR.

2.2.2. Simulation for heavy-tailed noise

In this simulation, we compare the performance of Wilcoxon change-point test (Dehling et al., 2015), CUSUM, simple neural network $\mathcal{H}_{L,\boldsymbol{m}}$ as well as truncated $\mathcal{H}_{L,\boldsymbol{m}}$ for heavy-tailed noise. Consider the model: $X_i = \mu_i + \xi_i, \quad i \geq 1$, where $(\mu_i)_{i\geq 1}$ are signals and $(\xi_i)_{i\geq 1}$ is a stochastic process. To test the null hypothesis

$$\mathbb{H} : \mu_1 = \mu_2 = \cdots = \mu_n$$

against the alternative

$$\mathbb{A} : \text{ There exists } 1 \leq k \leq n-1 \text{ such that } \mu_1 = \cdots = \mu_k \neq \mu_{k+1} = \cdots = \mu_n.$$

Dehling et al. (2015) proposed the so-called Wilcoxon type of cumulative sum statistic

$$T_n := \max_{1\leq k<n} \left| \frac{2\sqrt{k(n-k)}}{n} \frac{1}{n^{3/2}} \sum_{i=1}^{k} \sum_{j=k+1}^{n} \left(\mathbf{1}_{\{X_i < X_j\}} - 1/2 \right) \right| \tag{S3}$$

to detect the change-point in time series with outlier or heavy tails. Under the null hypothesis $\mathbb{H}$, the limit distribution of T_n‡ can be approximately by the supreme of standard Brownian bridge process $(W^{(0)}(\lambda))_{0\leq\lambda\leq 1}$ up to a scaling factor (Dehling et al., 2015, Theorem 3.1). In our simulation, we choose the optimal thresh value based on the training dataset by using the grid search method.

The truncated simple neural network means that we truncate the data by the z-score in data preprocessing step, i.e. given vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^\top$, then $x_i[|x_i - \bar{x}| > Z\sigma_x] = \bar{x} + \text{sgn}(x_i - \bar{x})Z\sigma_x$, $\bar{x}$ and σ_x are the mean and standard deviation of $\boldsymbol{x}$.

The training dataset is generated by using the same parameter settings of Figure 2(d) of the main text. The result of misclassification error rate (MER) of each method is reported in Figure S1. We can see that truncated simple neural network has the best performance. As expected, the Wilcoxon based test has better performance than the simple neural network based tests. However, we would like to mention that the main focus of Figure 2 of the main text is to demonstrate the point that simple neural networks can replicate the performance of CUSUM tests. Even though, the prior information of heavy-tailed noise is available, we still encourage the practitioner to use simple neural network by adding the z-score truncation in data preprocessing step.

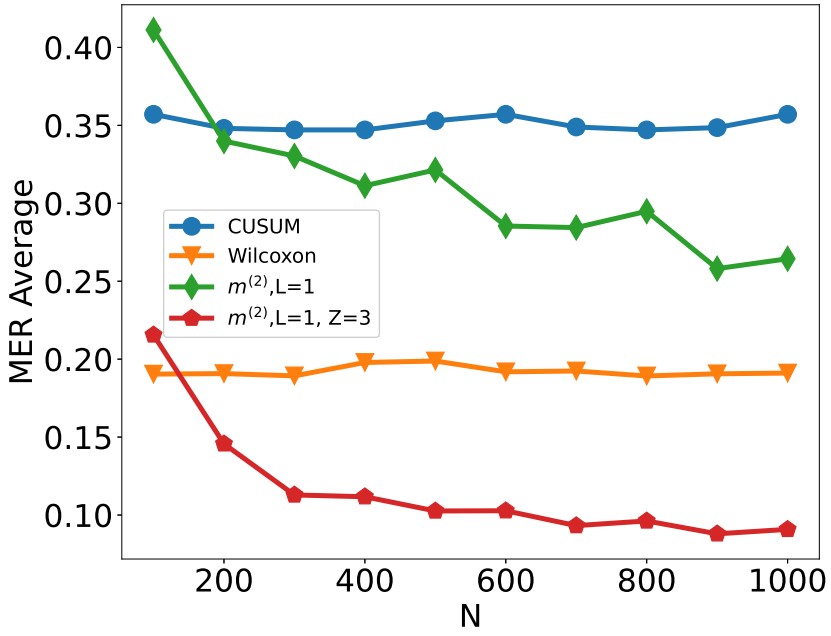


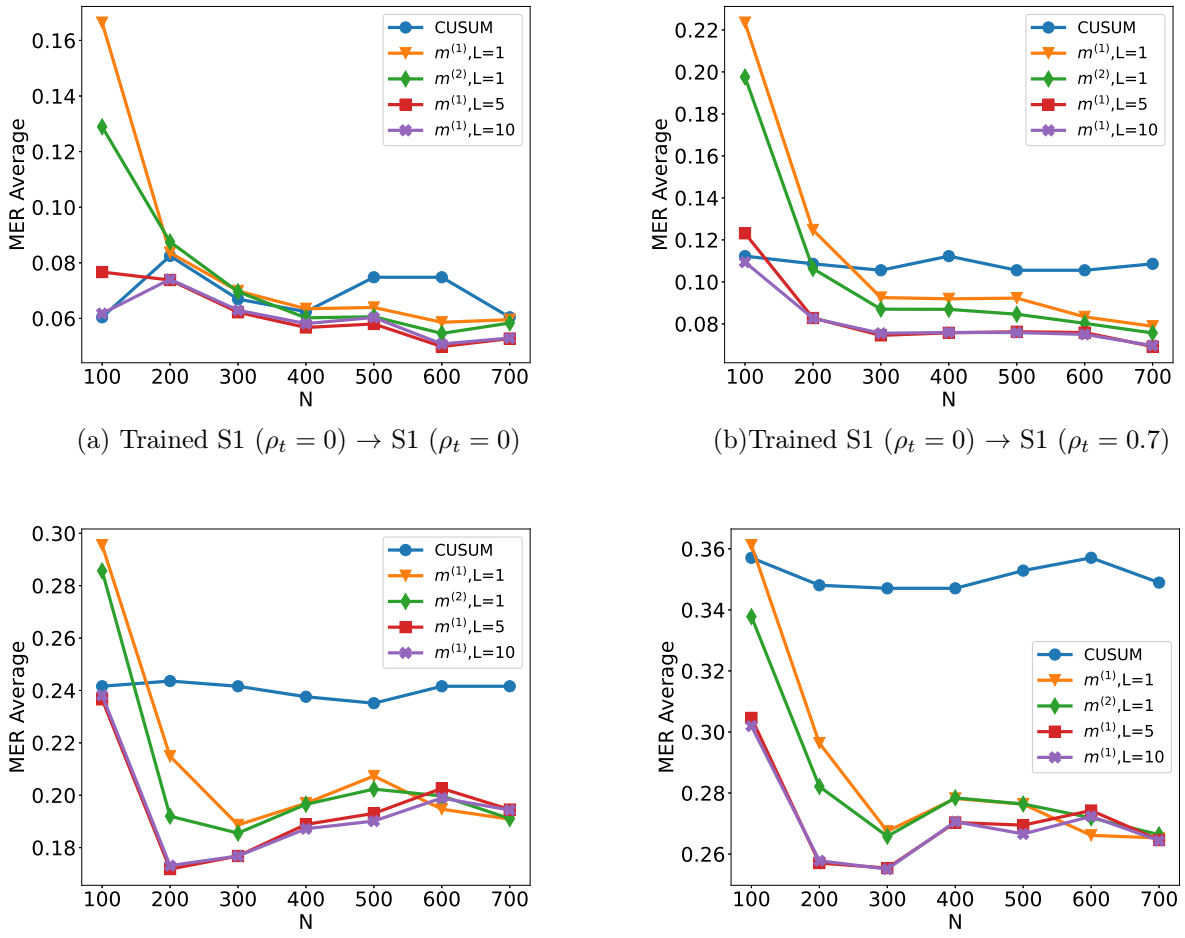
Fig. S1. Scenario S3 with Cauchy noise by adding Wilcoxon type of change-point detection method (Dehling et al., 2015) and simple neural network with truncation in data preprocessing. The average misclassification error rate (MER) is computed on a test set of size $N_{\text{test}} = 15000$, against training 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, Wilcoxon test, $\mathcal{H}_{1,m^{(2)}}$ and $\mathcal{H}_{1,m^{(2)}}$ with $Z = 3$ where $m^{(2)} = 2n - 2$ and $Z = 3$ means the truncated z-score, i.e. given vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^\top$, then $x_i[|x_i - \bar{x}| > Z\sigma_x] = \bar{x} + \text{sgn}(x_i - \bar{x})Z\sigma_x$, $\bar{x}$ and σ_x are the mean and standard deviation of $\boldsymbol{x}$.

2.2.3. Robustness Study

This simulation is an extension of numerical study of Section 5 in main text. We trained our neural network using training data generated under scenario S1 with $\rho_t = 0$ (i.e. corresponding to Figure 2(a)

‡The definition of T_n in Dehling et al. (2015, Theorem 3.1) does not include $2\sqrt{k(n-k)}/n$. However, the repository of the **R** package **robts** (Dürre et al., 2016) normalises the Wilcoxon test by this item, for details see function **wilcoxsuk** in here. In this simulation, we adopt the definition of (S3).

of the main text), but generate the test data under settings corresponding to Figure 2(a, b, c, d). In other words, apart the top-left panel, in the remaining panels of Figure S2, the trained network is misspecified for the test data. We see that the neural networks continue to work well in all panels, and in fact have performance similar to those in Figure 2(b, c, d) of the main text. This indicates that the trained neural network has likely learned features related to the change-point rather than any distributional specific artefacts.



(a) Trained S1 ($\rho_t = 0$) $\rightarrow$ S1 ($\rho_t = 0$)

(b)Trained S1 ($\rho_t = 0$) $\rightarrow$ S1 ($\rho_t = 0.7$)

(c) Trained S1 ($\rho_t = 0$) $\rightarrow$ S2

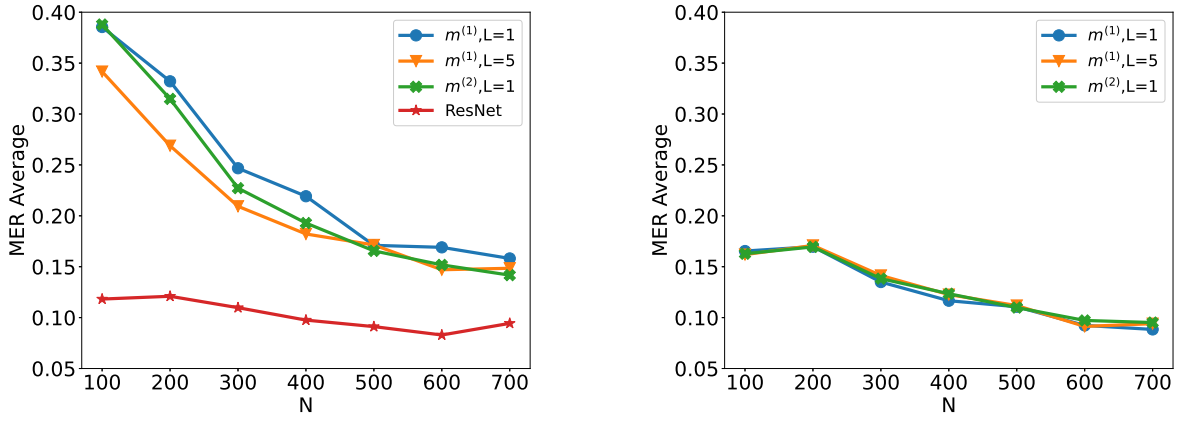
(d) Trained S1 ($\rho_t = 0$) $\rightarrow$ S3

Fig. S2. 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. The subcaption "A $\rightarrow$ B" means that we apply the trained classifier "A" to target testing dataset "B".

2.2.4. Simulation for change in autocorrelation

In this simulation, we discuss how we can use neural networks to recreate test statistics for various types of changes. For instance, if the data follows an AR(1) structure, then changes in autocorrelation can 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 deep 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, where we compare the performance of neural network based classifiers of various depths constructed with and without using

the transformed data as inputs.

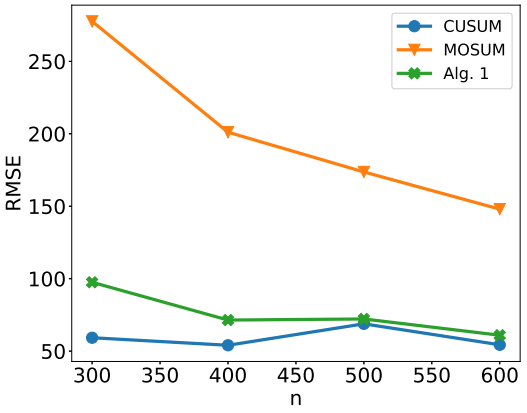


(a) Original Input (b) Original and $x_t x_{t+1}$ Input

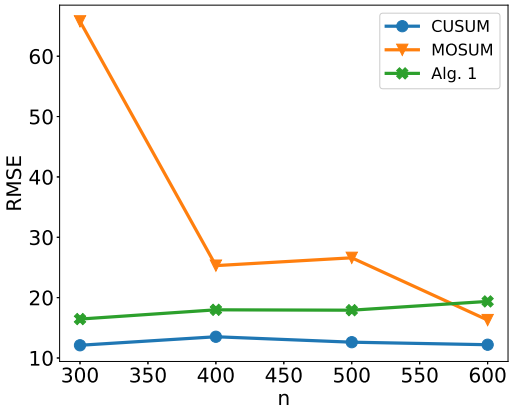
Fig. S3. Plot of the test set MER, computed on a test set of size $N_{\text{test}} = 30000$, against training sample size N for detecting the existence of a change-point on data series of length $n = 100$. We compare the performance of neural networks from four function classes: $\mathcal{H}_{1,m^{(1)}}$,$\mathcal{H}_{1,m^{(2)}}$, $\mathcal{H}_{5,m^{(1)}\mathbf{1}_5}$ and ResNet with 21 residual blocks where $m^{(1)} = 4\lfloor \log_2(n) \rfloor$ and $m^{(2)} = 2n - 2$ respectively. The change-points are randomly chosen from $\text{Unif}\{10, \ldots, 89\}$. Given change-point τ, data are generated from the autoregressive model $x_t = \alpha_t x_{t-1} + \epsilon_t$ for $\epsilon_t \overset{\text{iid}}{\sim} N(0, 0.25^2)$ and $\alpha_t = 0.2\mathbf{1}_{\{t<\tau\}} + 0.8\mathbf{1}_{\{t\geq\tau\}}$.

2.2.5. Simulation on change-point location estimation

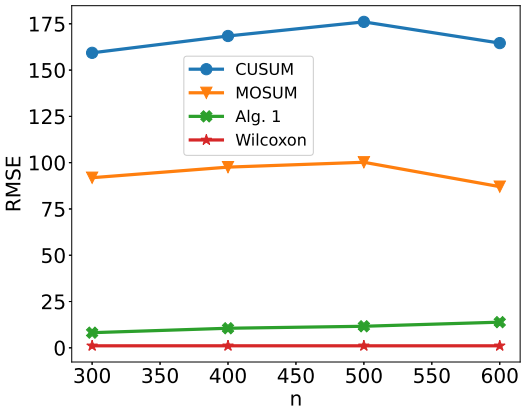
Here, we describe simulation results on the performance of change-point location estimator constructed using a combination of simple neural network-based classifier and Algorithm 1 from the main text. Given a sequence of length $n' = 2000$, we draw $\tau \sim \text{Unif}\{750, \ldots, 1250\}$. Set $\mu_L = 0$ and draw $\mu_R|\tau$ from 2 different uniform distributions: $\text{Unif}([-1.5b, -0.5b] \cup [0.5b, 1.5b])$ (Weak) and $\text{Unif}([-3b, -b] \cup [b, 3b])$ (Strong), 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 ratio. We then generate $\boldsymbol{x} = (\mu_{\text{L}} \mathbb{1}_{\{t\leq\tau\}} + \mu_{\text{R}} \mathbb{1}_{\{t>\tau\}} + \varepsilon_t)_{t\in[n']}$, with the noise $\boldsymbol{\varepsilon} = (\varepsilon_t)_{t\in[n']} \sim N_{n'}(0, I_{n'})$. We then draw independent copies $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N'}$ of $\boldsymbol{x}$. For each $\boldsymbol{x}_k$, we randomly choose 60 segments with length $n \in \{300, 400, 500, 600\}$, the segments which include τ_k are labelled '1', others are labelled '0'. The training dataset size is $N = 60N'$ where $N' = 500$. We then draw another $N_{\text{test}} = 3000$ independent copies of $\boldsymbol{x}$ as our test data for change-point location estimation. We study the performance of change-point location estimator produced by using Algorithm 1 together with a single-layer neural network, and compare it with the performance of CUSUM, MOSUM and Wilcoxon statistics-based estimators. As we can see from the Figure S4, under Gaussian models where CUSUM is known to work well, our simple neural network-based procedure is competitive. On the other hand, when the noise is heavy-tailed, our simple neural network-based estimator greatly outperforms CUSUM-based estimator.



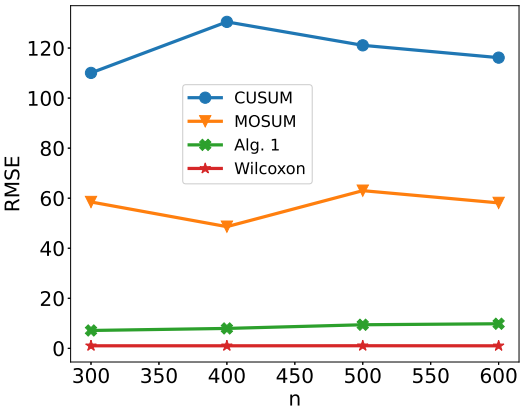
(a) S1 with $\rho_t = 0$, weak SNR



(b) S1 with $\rho_t = 0$, strong SNR



(c) S3, weak SNR



(d) S3, strong SNR

Fig. S4. Plot of the root mean square error (RMSE) of change-point estimation (S1 with $\rho_t = 0$ and S3), computed on a test set of size $N_{\text{test}} = 3000$, against bandwidth n for detecting the existence of a change-point on data series of length $n^* = 2000$. We compare the performance of the change-point detection by CUSUM, MOSUM, Algorithm 1 and Wilcoxon (only for S3) respectively. The RMSE here is defined by $\sqrt{1/N \sum_{i=1}^{N} (\hat{\tau}_i - \tau_i)^2}$ where $\hat{\tau}_i$ is the estimator of change-point for the i-th observation and τ_i is the true change-point. The weak and strong signal-to-noise ratio (SNR) correspond to $\mu_R|\tau \sim \text{Unif}([-1.5b, -0.5b] \cup [0.5b, 1.5b])$ and $\mu_R|\tau \sim \text{Unif}([-3b, -b] \cup [b, 3b])$ respectively.

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 3 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 change-point or no change-point existence in each time series. Next, we describe how to obtain this kind of training data from HASC sequential data. In general, Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^\top, d \geq 1$ be the d-channel vector. Define $\boldsymbol{X} := (\boldsymbol{x}_{t_1}, \boldsymbol{x}_{t_2}, \ldots, \boldsymbol{x}_{t_{n^*}})$ as a realization of d-variate time series where $\boldsymbol{x}_{t_j}, j = 1, 2, \ldots, n^*$ are the observations of $\boldsymbol{x}$ at n^* consecutive time stamps $t_1, t_2, \ldots, t_{n^*}$. Let $\boldsymbol{X}_i, i = 1, 2, \ldots, N^*$ represent the observation from the i-th subject. $\boldsymbol{\tau}_i := (\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,K})^\top, K \in \mathbb{Z}^+, \tau_{i,k} \in [2, n^* - 1], 1 \leq k \leq K$ 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 3 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 3 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 S7. 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) (see Figure S5). 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 S5.

- 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 (2015) and He et al. (2016).

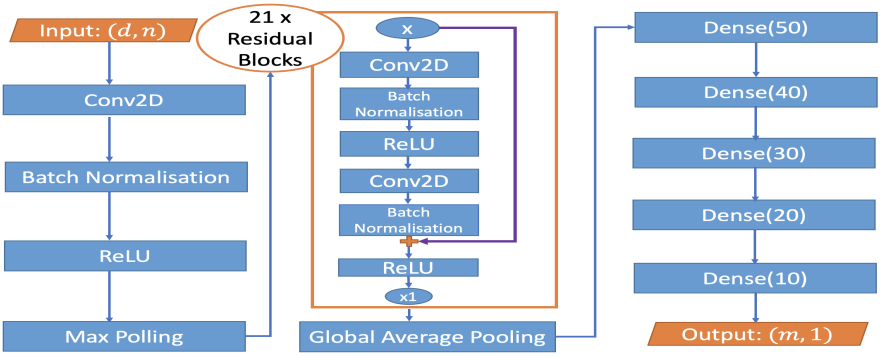


Fig. S5. Architecture of our general-purpose change-point detection neural network. The left column shows the standard layers of neural network with input size (d, n), d may represent the number of transformations or channels; We use 21 residual blocks and one global average pooling in the middle column; The right column includes 5 dense layers with nodes in bracket and output layer. More details of the neural network architecture appear in the supplement.

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 S5. 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 S6, we drop the dense layers "Dense(20)" and "Dense(10)" in Figure S5. 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 Section 6 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. S6. Label Dictionary

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

```
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. S7. Label Frequency

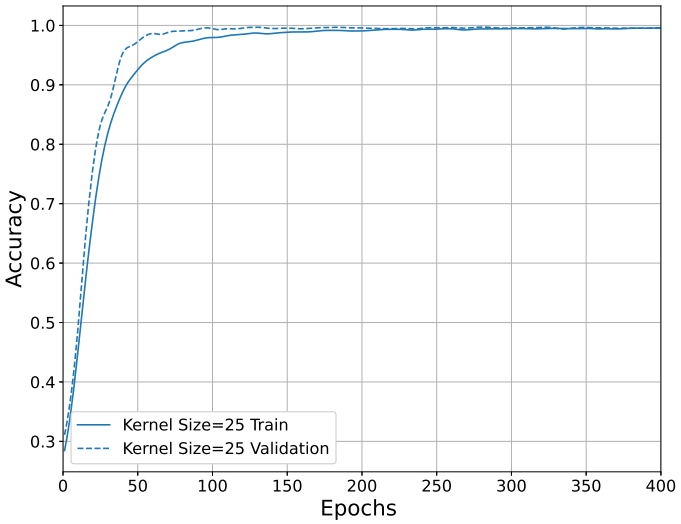


Fig. S8. The Accuracy Curves

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 S6.

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 S8 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 S9. 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. Figures S10 and S11 show the results of multiple change-point detection for other 2 sequential data sets from the 7-th person.

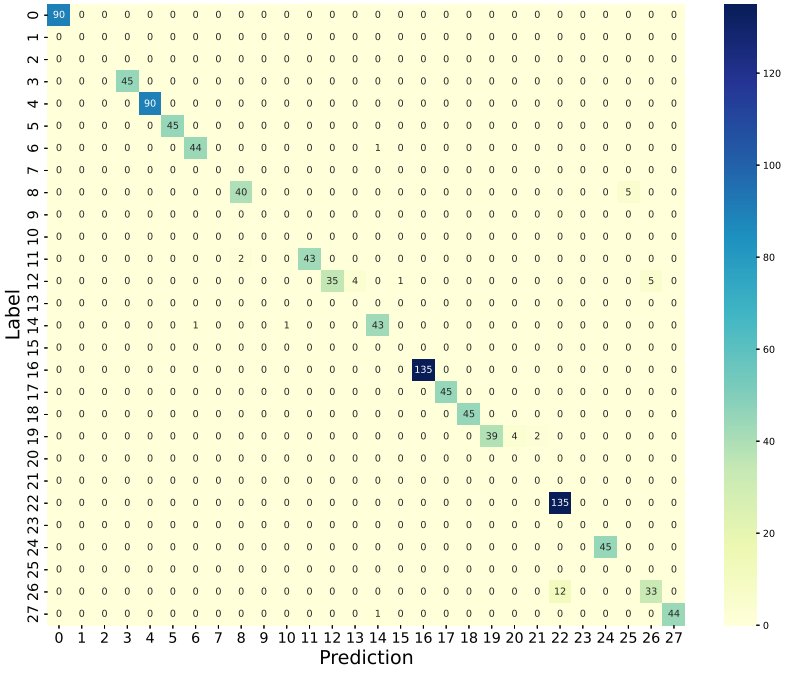


Fig. S9. Confusion Matrix of Real Test Dataset

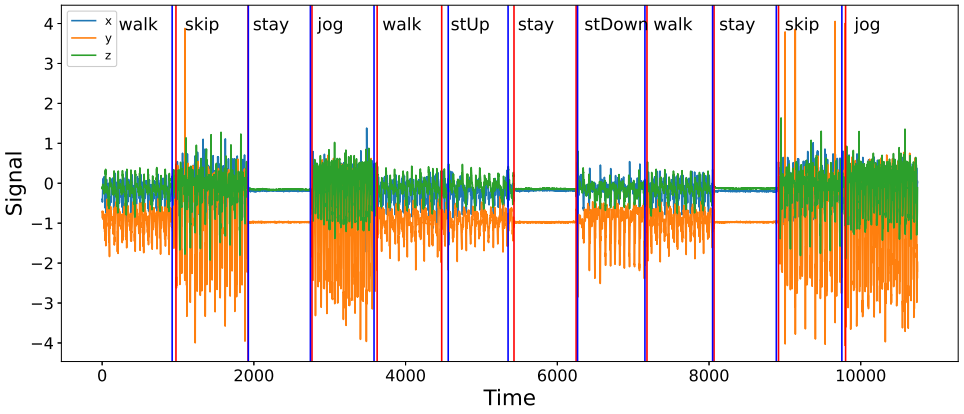


Fig. S10. Change-point Detection of Real Dataset for Person 7 (2nd sequence). The red line at 4476 is the true change-point, the blue line on its right is the estimator. The difference between them is caused by the similarity of “Walk” and “StairUp”.

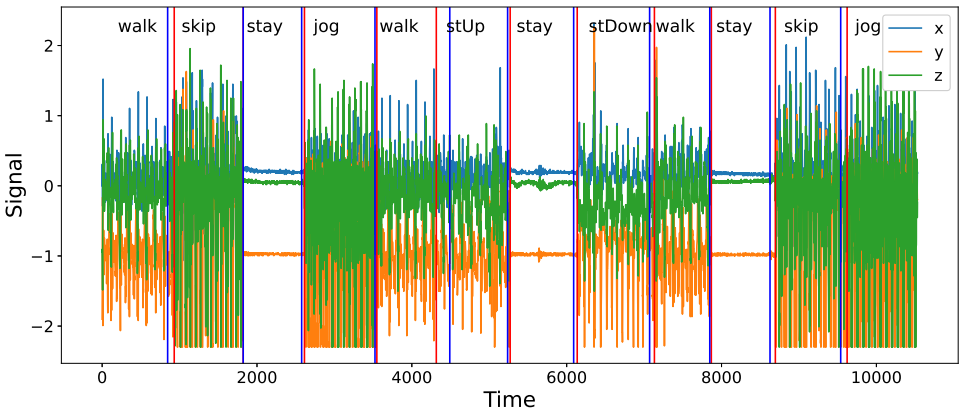


Fig. S11. Change-point Detection of Real Dataset for Person 7 (3rd sequence). The red vertical lines represent the underlying change-points, the blue vertical lines represent the estimated change-points.

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. *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.

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

Chen, J. and A. K. Gupta (2012). *Parametric Statistical Change Point Analysis: With Applications to Genetics, Medicine, and Finance* (2nd ed.). New York: Birkhäuser.

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.

Dürre, A., R. Fried, T. Liboschik, and J. Rathjens (2016). *robts: Robust Time Series Analysis*. R package version 0.3.0/r251.

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.

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 (2015). Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5353–5360.

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.

Ioffe, S. and C. Szegedy (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 448–456. JMLR.org.

Kuchibhotla, A. K. and A. Chakrabortty (2022). Moving beyond sub-Gaussianity in high-dimensional statistics: Applications in covariance estimation and linear regression. *Inf. Inference: A Journal of the IMA 11*(4), 1389–1456.

Li, J., P. Fearnhead, P. Fryzlewicz, and T. Wang (2022). Automatic change-point detection in time series via deep learning. *submitted*, arxiv:2211.03860.

Li, M., Y. Chen, T. Wang, and Y. Yu (2023). Robust mean change point testing in high-dimensional data with heavy tails. *arXiv preprint*, arxiv:2305.18987.

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.

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

Verzelen, N., M. Fromont, M. Lerasle, and P. Reynaud-Bouret (2020). Optimal change-point detection and localization. *arXiv preprint*, arxiv:2010.11470.

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.