AutoNet-Generated Deep Layer-Wise Convex Networks for ECG Classification

Yanting Shen[1*], Lei Lu[1,2*], Tingting Zhu[1], Xinshao Wang[1], Lei Clifton[3], Zhengming Chen[3], Robert Clarke[3], and David A. Clifton[1,4]

Abstract—The design of neural networks typically involves trial-and-error, a time-consuming process for obtaining an optimal architecture, even for experienced researchers. Additionally, it is widely accepted that loss functions of deep neural networks are generally non-convex with respect to the parameters to be optimised. We propose the Layer-wise Convex Theorem to ensure that the loss is convex with respect to the parameters of a given layer, achieved by constraining each layer to be an overdetermined system of non-linear equations. Based on this theorem, we developed an end-to-end algorithm (the AutoNet) to automatically generate layer-wise convex networks (LCNs) for any given training set. We then demonstrate the performance of the AutoNet-generated LCNs (AutoNet-LCNs) compared to state-of-the-art models on three electrocardiogram (ECG) classification benchmark datasets, with further validation on two non-ECG benchmark datasets for more general tasks. The AutoNet-LCN was able to find networks customised for each dataset without manual fine-tuning under 2 GPU-hours, and the resulting networks outperformed the state-of-the-art models with fewer than 5% parameters on all the above five benchmark datasets. The efficiency and robustness of the AutoNet-LCN markedly reduce model discovery costs and enable efficient training of deep learning models in resource-constrained settings.

Index Terms—AutoML, deep learning, deep neural networks, neural architecture search, layer-wise convex networks, electrocardiogram classification.

---

1 Introduction

Machine learning models have been increasingly used to analyze ECG signals, which are important clinical measurements for screening cardiovascular disease (CVD) [1], [2], [3]. Typically, convolutional neural networks (CNNs) [4], [5], residual blocks [3], [6], recurrent neural networks (RNNs) [7], [8], and transformer encoders [9], [10] are used as backbones to develop deep neural networks (DNNs) for feature extraction. Despite the remarkable performance of these deep learning models for ECG signal analysis, they are generally developed by trial-and-error, requiring substantial efforts and expertise in model design. Additionally, the randomness inherent in the training of neural networks due to random weight initialization, stochastic gradient estimation, and other sources of randomness makes model development particularly challenging [11], [12], as it is difficult to discern whether a change in performance is due to intervention (such as adding layers and changing hyperparameters) or due to randomness in training. Typically, researchers would train a model using the same set of hyperparameters on several occasions before concluding the benefits or hazards of an intervention. This process is undesirable for large models whose training process may take days or months.

There has been a growing interest in developing algorithmic solutions for neural architecture search (NAS) recently [13], [14], [15], [16]. NAS aims to introduce an efficient way to automate the process of developing deep learning models, putting an end to the trial-and-error practice of architecture design. Generally, there are three key components in an NAS framework: the architecture search space, module search strategy, and performance evaluation strategy [15], [16]. The core idea of NAS is to use a search strategy to find an optimal network structure in the predefined search space with limited computational cost [13], [14].

Early studies of NAS mostly used heuristic algorithms to drive the process of searching for architecture, such as reinforcement learning (RL) [17], [18] and evolutionary algorithms [19], [20]. These methods initially utilise a policy network to generate candidate architectures and evaluate them on a validation set. Then, the validation loss is used as a reward to update the policy network and train it to produce a more performant architecture [17], [18], [21]. However, these search methods often became computationally expensive, particularly when the task had a large search space. Recent NAS approaches employ elaborate strategies to speed up the search process, such as developing an expressive search space that supports complex topologies [19], integration of transfer learning and multi-objective evolution [22], weight-sharing one-shot architecture search [16], differentiable frameworks for block-wise architecture search [23], and knowledge distillation and adaptive combination of multiple searched networks [14].

- *DAC was supported by the Pandemic Sciences Institute at the University of Oxford; the National Institute for Health Research (NIHR) Oxford Biomedical Research Centre (BRC); an NIHR Research Professorship; a Royal Academy of Engineering Research Chair; the Wellcome Trust funded VITAL project (217650/Z/19/Z); and the InnoHK Hong Kong Centre for Centre for Cerebro-cardiovascular Engineering (COCHE). (*Yanting Shen and Lei Lu are the co-first authors of this work.)*
- [1]*Department of Engineering Science, University of Oxford, UK.*
- [2]*School of Life Course & Population Sciences, King's College London, London, UK.*
- [3]*Nuffield Department of Population Health, University of Oxford, UK.*
- [4]*Oxford Suzhou Centre for Advanced Research, Suzhou, China.*
- *Corresponding authors: Lei Lu, email: lei.lu@eng.ox.ac.uk; Xinshao Wang, email: xinshaowang@gmail.com.*

NAS has demonstrated advancements in improving model performance across various applications, such as image processing [24], [25], semantic segmentation [26], [27], and object detection [28], [29]. Recent research also explores the use of NAS for healthcare applications, such as electroencephalography (EEG) data processing [30], muscle fatigue detection [31], cardiac abnormality diagnosis [32], and heartbeat classification [33]. Moreover, an NAS was developed by leveraging k-fold cross-validation, and the deep learning model was evaluated on data from the UCR archive [34]. However, the development of NAS still faces significant limitations. Searching through every possible architecture, one of the most fundamental approaches of NAS, is computationally prohibitive, which requires vast resources and time. While algorithms like RL reduce the need for exhaustive search, they use a defined space of operations, limiting the potential to discover more efficient or effective designs that fall outside the search space. Additionally, the majority of these search strategies are treated as a black-box optimisation problem [13], [14], [16], [21], [22], which necessitates a large number of architecture evaluations, and it is also challenging to explain why these approaches lead to model performance improvement.

The development of automated ECG analysis is critical in cardiovascular medicine, where the ECG signal has been a long-standing source of valuable insights and cost-effective solutions for managing cardiovascular diseases (CVDs) [7], [8], [35]. Examples include using large sets of ECGs to develop deep learning models for predicting atrial fibrillation [1], ventricular dysfunction [3], myocardial infarction [5], and heart failure [36], as well as assessing mortality risks [37]. While these studies have demonstrated promising results for deep learning in ECG analysis, the models are typically designed empirically, relying on hand-crafted building blocks, which are highly sensitive to the choice of feature extractors. In this context, NAS offers the potential to create an optimal model that could improve healthcare outcomes and enable the generalisation of the model for diverse healthcare applications.

In this work, we propose a novel NAS framework to generate optimal deep learning models for automated ECG data analysis. In particular, we propose the Layer-Wise Convex Networks (LCNs) that enable us to search for optimal models based on the characteristics of the training set. We begin by providing an overview of the core principles of deep learning, followed by the derivation of our proposed LCNs and a theorem with the same name. We then introduce AutoNet, a heuristic algorithm designed to automatically generate deep LCNs based on the characteristics of the training set. Finally, we demonstrate the performance of auto-generated LCNs by comparing them to the state-of-the-art deep learning model for ECG classification on three datasets: (i) International Conference on Biomedical Engineering and Biotechnology (ICBEB)[1] Physiological Signal Challenge 2018, (ii) the PhysioNet Atrial Fibrillation Detection Challenge 2017 [38], and (iii) the China Kadoorie Biobank (CKB)[2]. To assess the generalisation of our model, we further validated the proposed AutoNet algorithm on non-ECG datasets.

1. http://2018.icbeb.org/Challenge.html
2. https://www.ckbiobank.org/site/

The contributions of this paper are:

- We propose an efficient AutoNet-LCN algorithm that automatically determines the optimal architecture of the deep neural networks for customised datasets and applications.
- Instead of a vast search space for learnable parameters of deep learning models, our proposed LCN theorem can reduce the search space of NAS to one dimension.
- We provide compelling evidence to validate the effectiveness of our developed NAS in the relatively unexplored yet highly essential realm of time-series data analysis.
- This is the first time that an NAS framework has been benchmarked on multiple healthcare datasets with variations in data durations, signal-to-noise ratios, number of classes, and sampling frequencies.

The remainder of this paper is organised as follows. Section 2 describes the notations that are used for the model developed. Section 3 presents our proposed LCN theorem, and Section 4 develops the AutoNet algorithm for model generation. Section 5 presents the experiments and their results using three ECG datasets (ICBEB, PhysioNet, a private dataset from CKB), and additional validation on two non-ECG datasets. Section 6 discusses the strengths and limitations of this study. Finally, conclusions are drawn in Section 7.

2 NOTATIONS

Without loss of generality, we introduce the notations through a K-class classification problem. Let $\boldsymbol{X} \in \mathbb{R}^{D\times m}$ denote the design matrix, where D is the dimension of the feature vector, and m is the number of training examples. $\boldsymbol{Y} \in \mathbb{R}^{K\times m}$ represents the training targets, where K is the number of classes. Let $\hat{\boldsymbol{Y}}$ represent the prediction of $\boldsymbol{Y}$ given by an L-layer neural network. Then, each layer of the network computes:

$$\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]}\boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}, \tag{1}$$

$$\boldsymbol{A}^{[l]} = g^{[l]}(\boldsymbol{Z}^{[l]}), \tag{2}$$

where, $l = 0, 1, ..., L$ is the layer index, with 0 and L representing the input and output layers, respectively. In other words, $\boldsymbol{A}^{[0]} = \boldsymbol{X}$, and $\boldsymbol{A}^{[L]} = \hat{\boldsymbol{Y}}$. The matrix $\boldsymbol{A}^{[l]} \in \mathbb{R}^{n^{[l]}\times m}$ is referred to as the *activation* or *output* of layer l. The function $g^{[l]}$ typically denotes the non-linear activation function of layer l. $\boldsymbol{Z}^{[l]} \in \mathbb{R}^{n^{[l]}\times m}$ represents the affine transformation of the activations of layer $l-1$. The matrix $\boldsymbol{W}^{[l]} \in \mathbb{R}^{n^{[l]}\times n^{[l-1]}}$ denotes the weight matrix connecting layer $l-1$ to layer l in the forward pass, where $n^{[l-1]}$ and $n^{[l]}$ represent the number of neurons in layers $l-1$ and l, respectively. $\boldsymbol{b}^{[l]} \in \mathbb{R}^{n^{[l]}}$ denotes the bias vector of layer l.

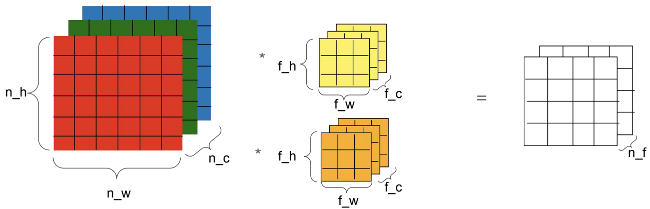


Figure 1: The convolution operation of a filter

We illustrate the notations of CNNs in Fig. 1 and formulate the calculations in equations 3-5. Each of the small yellow patches in Fig. 1 is referred to as a *kernel* or *filter*.

$$z_{11} = w_{111}x_{111} + w_{121}x_{121} + ... + w_{333}x_{333}, \tag{3}$$

$$z_{12} = w_{111}x_{121} + w_{121}x_{131} + w_{131}x_{141} + ... + w_{333}x_{343}, \tag{4}$$

$$z_{i',j'} = b + \sum_{k=1}^{f_c}\sum_{i=1}^{f_h}\sum_{j=1}^{f_w} w_{i,j,k}x_{(i'-1)s+i,(j'-1)s+j,k}, \tag{5}$$

where, b represents the bias parameter, with one bias per filter by convention. c denotes the number of input channels, while f_h and f_w indicate the kernel's height and width, respectively. w denotes the kernel weight, and x represents the element in the input tensor. s denotes the stride, and p represents the padding hyperparameter. It is worth noting that the input tensor and the filters must have the same number of channels.

Let f_h, f_w, and f_c denote the height, width, and number of channels of the filter, respectively. Then, the filter has $f_h \times f_w \times f_c$ weight parameters. The resulting tensor from the convolution operation, denoted as $\boldsymbol{Z} = z_{i',j'}$, is called a *feature map*. If we have n_f filters, then we will have n_f feature maps. Following the convention of having one bias parameter per filter, a convolutional layer with n_f filters has $n_f \times f_h \times f_w \times f_c$ weights and n_f bias parameters. The filters in CNN are equivalent to the *neurons* in feed-forward neural networks.

In summary, if the dimension of input tensor to a convolutional layer is $n_h \times n_w \times n_c$, the kernel dimension is $f_h \times f_w \times f_c$, and there are n_f filters, and we use the convention of one bias per filter, and pad p rows or columns on all edges, with stride s, and by convention $f_c = n_c$; Then, the output shape of such a convolutional layer is $\lfloor 1 + \frac{n_h - f_h + 2p}{s} \rfloor \times \lfloor 1 + \frac{n_w - f_w + 2p}{s} \rfloor \times n_f$, and the number of parameters (weights and biases) is $n_f \times (f_h \times f_w \times f_c + 1)$.

3 Layer-Wise Convex Networks

3.1 Motivation

The Layer-Wise Convex Network (LCN) theorem is motivated by the rational and effective design of neural networks using the training dataset. A feed-forward neural network is essentially a computational graph where each layer can only "see" the layers directly connected to it, and has no way to tell whether its upstream layer is an input layer or a hidden layer. This "layer-unawareness" idea is similar to what is acknowledged in the development of batch normalisation [39] and is the central idea of the LCN theorem. The LCN approaches machine learning from function approximation and information theory perspectives.

3.2 The Layer-Wise Convex Theorem

Theorem 1. *For an L-layer feed-forward neural network, the sufficient conditions for there existing a unique set of parameters $\boldsymbol{W}^{[l]}$ and $\boldsymbol{b}^{[l]}$ that minimises the Euclidean distance $|\boldsymbol{A}^{[l]} - g^{[l]}(\boldsymbol{W}^{[l]}\boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]})|_2, \forall l \in [1, L]$ are:*

- *$n_W^{[l]} + n_b^{[l]} \le m$, $\forall l \in [0, L]$, where m is the number of training examples, and $n_W^{[l]}$ and $n_b^{[l]}$ are the number of weights and biases in layer l, respectively.*
- *The network does not have skip connections.*
- *All activation functions of the network are strictly monotonic, but different layers may have different monotonicity. For example, some layers can be strictly increasing, while other layers can be strictly decreasing.*
- *All reverse functions of the activation functions are Lipschitz continuous.*

Definition 3.1. Layer-Wise Convex Network: Any network fulfilling Theorem 1 is called a Layer-Wise Convex Network (LCN).

3.3 Sketch of Proof

Suppose we have a training set $\boldsymbol{X} \in \mathbb{R}^{D\times m}$ with training labels $\boldsymbol{Y} \in \mathbb{R}^n$, and there exists a deterministic data generating process $f : \boldsymbol{X} \mapsto \boldsymbol{Y}$. Our aim is to approximate the data generating process f using a neural network. The universal approximation theorem [40], [41] states that a feed-forward neural network with a linear output and at least one sufficiently wide hidden activation layer with a broad class of activation functions, including sigmoid and piecewise linear functions [42], can approximate any continuous function and its derivative defined on a closed and bounded subset of $\mathbb{R}^n$ to arbitrary precision [43]. According to the universal approximation theorem, there exists a set of neural network parameters $\boldsymbol{\theta}$ such that

$$|f - f(\boldsymbol{\theta})| < \epsilon, \tag{6}$$

$\forall \epsilon > 0$. As the neural network computes a chain of functions, if we can find $\boldsymbol{\theta}$, then we have the following equations:

$$|g^{[l]}(\boldsymbol{\theta}^{[l]}\tilde{\boldsymbol{A}}^{[l-1]}) - \tilde{\boldsymbol{A}}^{[l]}| < \epsilon, \tag{7}$$

$$\boldsymbol{A}^{[0]} = \boldsymbol{X}, \tag{8}$$

$$|\boldsymbol{A}^{[L]} - \boldsymbol{Y}| < \epsilon, \tag{9}$$

where, $l \in [0, L]$ is the layer index; $\tilde{\boldsymbol{A}}^{[l]} \in \mathbb{R}^{(n^{[l]}+1)\times m}$ and it differs from $\boldsymbol{A}^{[l]}$, as it has one dummy row of 1s to include $\boldsymbol{b}$ into $\boldsymbol{\theta}$; in other words, $\tilde{\boldsymbol{A}} = [\boldsymbol{1}; \boldsymbol{A}]$. To estimate $\boldsymbol{\theta}$, recall an over-determined system of linear equations $\boldsymbol{Ax} = \boldsymbol{y}$ has a unique set of solutions that minimises the Euclidean distance $|\boldsymbol{Ax} - \boldsymbol{y}|_2$. This property also extends to nonlinear equations, as long as the nonlinear activation $g^{[l]}$ is strictly monotonic and its reverse function is Lipschitz continuous.

Formally, a real function h is said to be Lipschitz continuous if one can find a positive real constant K such that

$$|h(x_1) - h(x_2)| \leq K|x_1 - x_2|, \tag{10}$$

for any real x_1 and x_2 in the domain of h. Any function with a bounded gradient on its domain is Lipschitz continuous. As the inverse of a strictly monotonic function is defined and unique, we write the equivalent form of inequality (7) by taking inverse on both sides,

$$g^{-1[l]}(\tilde{\boldsymbol{A}}^{[l]} - \epsilon) < \boldsymbol{\theta}^{[l]}\tilde{\boldsymbol{A}}^{[l-1]} < g^{-1[l]}(\tilde{\boldsymbol{A}}^{[l]} + \epsilon). \tag{11}$$

Using Lipschitz continuity of $g^{-1[l]}$, we can find a positive real constant K such that

$$\begin{aligned} g^{-1[l]}(\tilde{\boldsymbol{A}}^{[l]}) - K\epsilon \leq g^{-1[l]}(\tilde{\boldsymbol{A}}^{[l]} - \epsilon) < \\ \boldsymbol{\theta}^{[l]}\tilde{\boldsymbol{A}}^{[l-1]} < g^{-1[l]}(\tilde{\boldsymbol{A}}^{[l]} + \epsilon) \leq g^{-1[l]}(\tilde{\boldsymbol{A}}^{[l]}) + K\epsilon, \end{aligned} \tag{12}$$

$\forall \epsilon > 0$, which implies

$$|\boldsymbol{\theta}^{[l]}\tilde{\boldsymbol{A}}^{[l-1]} - g^{-1}(\tilde{\boldsymbol{A}}^{[l]})| < K\epsilon, \tag{13}$$

$$\lim_{\epsilon \to 0} \boldsymbol{\theta}^{[l]}\tilde{\boldsymbol{A}}^{[l-1]} = g^{-1}(\tilde{\boldsymbol{A}}^{[l]}). \tag{14}$$

We showed the estimation of $\boldsymbol{\theta}$ at the l^{th} layer in equation (14), $l = 1, 2, \cdots, L$, which uses Lipschitz continuity to transform the inequality (7) into a set of linear equations. It can be seen from equation (14) that, in an ideal case, there is a unique and optimal solution $\boldsymbol{\theta}^{[l]}$ for each layer l when the number of equations $n_{\boldsymbol{\theta}}$ is equal the number of training samples m. However, the condition is too harsh when designing neural networks, therefore, we generalise the constraint as the inequality (7); and if the system is overdetermined, i.e., $n_{\boldsymbol{\theta}} \leq m$, we can always find a set of parameters to minimise the Euclidean distance for the estimation. In practice, as shown in Section 4, we find the maximum number of parameters that are close to the number of training samples, which will make the solution have a sufficiently small error for the estimation.

3.4 Relaxation of the LCN Constraints

The strict monotonicity of activations and the no-skip connections are necessary to prove the uniqueness of the solution to the system of non-linear equations between each layer. However, the LCN theorem does not consider the problem of gradient vanishing for very deep neural networks; therefore, the resulting architecture may theoretically have appropriate model capacity, but cannot be trained effectively. With this in mind, we relax the condition of no-skip connections, and allow for skip connections when the model is too deep to be trained effectively. We investigate the effectiveness of relaxing LCN constraints in Section 5.

In addition, it is of interest to study whether the monotonicity condition of LCN can be relaxed and allows for non-strictly monotonic activation functions (e.g., ReLU). This would enable the application of the LCN theorem to a variety of neural networks. Therefore, we compare two variants of LCN, ReLU-LCN and Leaky-LCN in the Section 5, where the hidden layer activations of ReLU-LCN are all ReLU, and the hidden layer activations of Leaky-LCN are all leaky ReLU with $\alpha = 0.3$, denoted as follows,

$$y = \begin{cases} x & \text{if } x > 0, \\ \alpha x & \text{if } x \leq 0. \end{cases} \tag{15}$$

4 AutoNet Algorithm for Searching LCNs

We design the AutoNet algorithm that allows us to automatically generate a layer-wise convex network with a given dataset, outlined in Algorithms 1 and 2.

4.1 Timescale Hyperparameter for Sequential Inputs

In this study, we simply the process of designing neural networks by using repeated feature learning blocks. For example, each block of the stack convolutional layers is followed by a max-pooling layer [1]. Then, the key question of the estimation is how to calculate the number of repeated blocks. We are motivated by the fact that many time-series signals have the property of periodicity, informing that the timescale of the period can be helpful for the model to learn latent features from the periodic data. We therefore use the term $f_s\tau$ to estimate the repeated model blocks, where f_s is the sampling frequency, and τ is a timescale parameter for the rough estimation of periodicity. Then, the number of max-pooling layers is estimated as follows,

$$n_{maxpool} = \lceil \log_p(f_s\tau) \rceil, \tag{16}$$

For example, if the input time-series ECG data has a sampling frequency of 500Hz , the timescale $\tau = 1s$, and p is the pooling size with a default value $p = 2$, then we can calculate the number of max-pooling layers as $\lceil \log_2(1 \times 500) \rceil = 9$. If the input signal is not apparently periodic, we can set $d = f_s\tau$ and roughly estimate the value to be the length of the entire or half of input time-series data.

4.2 An Example of Generating the Baseline LCN Model

We present an example of using the LCN theorem to design model architecture for the CKB dataset, which is a four-class classification task. Each training example is a 12-lead ECG time series with a 10s time duration and 500Hz sampling frequency, thus the input dimension D of each training example is $500 \times 10 \times 12 = 60,000$. According to the LCN theorem, the number of parameters per layer should not exceed the number of training samples ($n_{sample} = 6,065$). Because $D > m$, if we use a feed-forward network, the first layer will have at least D parameters, then we must use weight-sharing mechanisms; Meanwhile, because we are analysing time-series data, 1-D CNN is a natural choice. In this work, we use 1-D CNN with the conventional parameter of $n_h = 1$, and f_h is also constrained to be 1.

We design the networks using repeated structures, ensuring that all layers maintain the same output shape until the final output layer. This repeated structure not only reduces the number of hyperparameters but also mitigates the issues of gradient vanishing or exploding [44]. It is recommended to avoid adding fully connected layers between the last convolutional layer and the output layer to prevent exceeding the upper bound. The dimension of densely connected layers has to be very small, which means that it will become "bottlenecks" in the flow of information. Therefore, we utilize only convolutional, pooling (for dimension reduction), and softmax output layers. When using a CNN layer with kernel size k, stride s, padding p, and the number of filters n_f, the output shape of the convolutional layer is $(\lfloor\frac{input\ dimension-k+2p+1}{s}\rfloor, n_f)$, and the number of parameters for this layer is $n_f(kn_f+1)$ (assuming multiple convolutional layers are stacked together). Since a stride $s>1$ results in dimension reduction and empirically worse performance than max-pooling, we maintain $s=1$. To keep the output shape identical to the input shape, we set the parameter as "same" padding, then we calculate k and n_f as follows,

$$k = n_f = \operatorname{argmax} n_f(n_f^2+1), \tag{17}$$

subject to

$$n_f(n_f^2+1) \leq m. \tag{18}$$

We constrain $k=n_f$ to avoid k being unreasonably large for long signals with few channels.

After calculating the hyperparameters k and n_f, and obtaining the number of max-pooling layers from equation (16), we are able to develop the baseline model (Fig. 2) for the CKB dataset. We stack convolutional layers between max-pooling layers for model generation. The number of convolutional layers stacked between max-pooling layers is a hyperparameter, denoted as n_{repeat}. The next step is to determine the depth of the deep neural networks. However, there is no guideline for calculating the optimal depth; the principle is that adding more layers should not harm the model performance.

4.3 AutoNet for Deep Neural Network Generation

We note that the width and depth of convolutional layers are two important hyperparameters in developing deep neural networks. In this study, we introduced the LCN Theorem to calculate the width of the neural networks, and proposed a hierarchical approach AutoNet (Algorithms 1 and 2) to search the depth of the model. Combining the two parts, the method allows us to automatically search the architecture of the deep LCNs. Particularly, in Algorithm 1, we calculate the width of the neural networks according to Theorem 1, and then generate a baseline model LCN; In Algorithm 2 we update the LCN model by increasing the value of n_{repeat}, and we track the losses of training and validation. The parameter n_{repeat} will stop increasing when neither of them decreases. Next, skip connections and batch normalisation will be added to the building blocks, which attempt to improve the gradient flow for model training. We describe our proposed AutoNet for the generation of deep neural networks with the following steps.

4.3.1 *Step One: Generate the Baseline Model*

The LCN model for ECG classification has only five hyperparameters: $n_{repeat} \in \mathbb{N}$, $n_{maxpool} \in \mathbb{N}$, $n_f \in \mathbb{N}$, $skip \in \mathbb{B}$ (Boolean domain), and $bn \in \mathbb{B}$, which can be determined by the training set and the AutoNet algorithm. n_f is the number of filters of each convolutional layer, calculated according to the LCN theory using the number of whole training samples. The number of max-pooling is determined by equation (16). The output layer of the model is a time-distributed softmax layer, which classifies the entire signal by majority voting. After calculating the hyperparameters, the baseline LCN model is trained using Algorithm 1 over mini-batches. The parameters $skip$ and bn are the "switches" indicating whether the network adds skip connections and batch normalisation, respectively.

Algorithm 1: Build a LCN. See Fig. 3 for the positions of convolutional, activation, batch normalisation, and maxpooling layers.

Input: m, $n_{channel}$, n_{class}, n_{repeat}, $skip$, bn, $n_{maxpool}$.
Output: model.

$n_f = \operatorname{argmax}_{n_f} n_f(n_f^2+1)$ subject to $n_f(n_f^2+1) \leq m$.
add the input layer.
if bn **then**
  add a batch normalisation layer.
end
add a convolutional layer, kernel size $= n_f$, $n_{filters} = n_f$.
if bn **then**
  add a batch normalisation layer.
end
add a maxpooling layer, pooling size$=2$.
for _ *in range* $n_{maxpool}-1$ **do**
  for _ *in range* n_{repeat} **do**
    add a convolutional layer, kernel size $= n_f$, $n_{filter} = n_f$.
    if $skip$ **then**
      connect the before-activation output of every $n_{maxpool}-1$ convolutional layers by addition.
    end
    add an activation (ReLU or leaky ReLU) layer.
    if bn **then**
      add a batchnorm layer.
    end
  end
  add a maxpooling layer.
end
add a time-distributed softmax layer.

4.3.2 *Step Two: Develop the Model*

With the developed baseline LCN model, we use Algorithm 2 to generate the optimal deep neural networks for the classification task, which is outlined as follows.

- Start with the baseline model, without batch normalisation nor skip connection, i.e., $bn = FALSE, skip = FALSE$, and $n_{repeat} = 1$. The stopping criterion is no reduction in validation loss for eight epochs.
- Increase n_{repeat} by one each time, until *neither the training loss nor the validation loss* decreases, then turn on skip connection and connect every

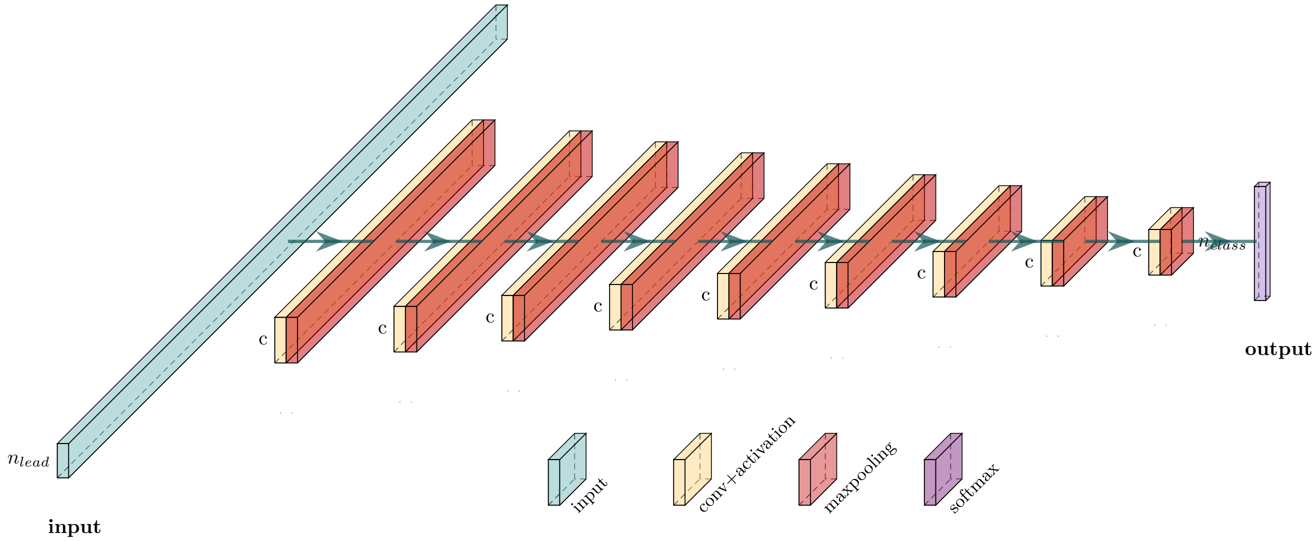


Figure 2: Baseline model architecture. The number of max-pooling layers is calculated by equation (16). Before each max-pooling layer, the baseline model has one convolutional layer and one activation layer, which can be ReLU or Leaky ReLU. When adding skip connections, the post-convolution (before activation) tensor is added to every $n_{maxpool} - 1$ post-convolution tensor (see Fig. 3). When necessary, the batch normalisation layers are added after the input layer and each activation layer.

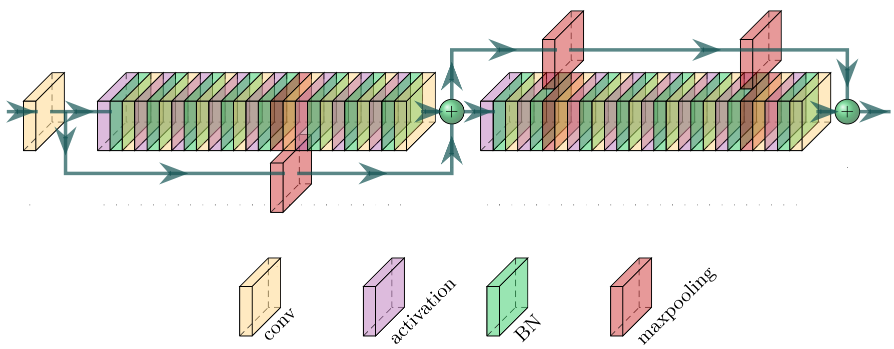


Figure 3: The positions of convolutional, activation, batch normalisation, max-pooling layers, and the skip connection. The illustrated network has a repeated structure of convolution-activation-BN, with $n_{maxpool} = 9$, $n_{repeat} = 5$. The max-pooling layer is added after every n_{repeat} (5 in this example) batch normalisation layers. The element-wise addition is applied to the output tensor of every $n_{maxpool} - 1$ (8 in this example) convolutional layers. For example, the output tensor of the first convolutional layer is element-wisely added to the output tensor of the 9th convolutional layer, and the resulting tensor is the input to the following activation layer, which is also used in the element-wise addition with the output tensor of the 17th convolutional layer.

$n_{maxpool} - 1$ layer by adding the post-convolution-before-activation tensors with the output tensor of $n_{maxpool} - 1$ convolutional layers (Fig. 3).

- Increase n_{repeat} by one each time, until *neither the training loss nor the validation loss* decreases, then add batch normalisation after each activation layer.
- Increase n_{repeat} by one each time until *neither the training loss nor the validation loss* decreases. The model which yields minimum validation loss is selected to be the "best" model.

4.3.3 Step Three: Model Averaging

We first train the identified "best" network architecture K times, yielding K models. Then, we calculate the average probability predictions provided by these K models to classify the case into the class with the highest mean probability, i.e.,

$$\hat{i} = \underset{i}{\operatorname{argmax}} \frac{1}{K} \sum_{j=1}^{K} p_{ij}, \tag{19}$$

where p_{ij} is the probability of i-th class predicted by the j-th model. This step can be omitted if one is not reporting the final results and intends to prototype quickly.

5 Experiments and Results

We compare LCN models generated by our AutoNet with Hannun-Rajpurkar's ResNet model [1]. The latter has been

Algorithm 2: Develop the model using AutoNet. This algorithm calls Algorithm 1 to build each LCN, then train the model until early stopping criteria is met. It tracks the minimum training loss and the minimum validation loss during training and compare them against the policy.

Input: m, $n_{channel}$, n_{class}, n_{repeat}, $skip$, bn, $n_{maxpool}$, $\boldsymbol{X}$, $\boldsymbol{Y}$, model_averaging, $fold = 10$.
Output: best model.

batch size = 32, patience = 8, bn = False, $skip$ = False.
build a LCN model using Algorithm 1 and train it.
while *min_train_loss or min_validation_loss declines* **do**
  $n_{repeat} = n_{repeat} + 1$.
  build a new LCN using Algorithm 1 and train it.
  update min_train_loss and min_validation_loss.
end
$skip$ =True.
while *min_train_loss or min_validation_loss declines* **do**
  $n_{repeat} = n_{repeat} + 1$.
  build a new LCN using Algorithm 1 and train it.
  update min_train_loss and min_validation_loss.
end
bn =True.
while *min_train_loss or min_validation_loss declines* **do**
  $n_{repeat} = n_{repeat} + 1$.
  build a new LCN using Algorithm 1 and train it.
  update min_train_loss and min_validation_loss.
end
best_model = the model with min_validation_loss.
if *model_average* **then**
  train the best network $fold$ times.
  best_model = the average ensemble of the $fold$ models.
end

demonstrated to exceed average cardiologist performance in classifying 12 rhythm classes on 91,232 recordings, and is regarded as the state-of-the-art.

5.1 ECG Datasets

5.1.1 ICBEB Dataset

The publicly available training set of the International Conference on Biomedical Engineering and Biotechnology (ICBEB) 2018 challenge includes 12-lead 500Hz 5-143s ECG time-series waveform from 6,877 participants (3,178 female and 3,699 male). The dataset has nine classes. The primary evaluation criterion of the Challenge is the 9-class average F_1 score, and the secondary evaluation criteria are F_1 scores of sub-abnormal classes: F_{AF}, F_{Block}, F_{PC}, F_{ST}, which are calculated as follows [45],

$$F_1 = \frac{1}{9}\sum_{i=1}^{9} \frac{2N_{ii}}{\sum_{j=1}^{9}(N_{ij}+N_{ji})}, \tag{20}$$

$$F_{AF} = \frac{2N_{22}}{\sum_{j=1}^{9}(N_{2j}+N_{j2})}, \tag{21}$$

$$F_{Block} = \frac{2\sum_{i=3}^{5} N_{ii}}{\sum_{i=3}^{5}\sum_{j=1}^{9}(N_{ij}+N_{ji})}, \tag{22}$$

$$F_{PC} = \frac{2\sum_{i=6}^{7} N_{ii}}{\sum_{i=6}^{7}\sum_{j=1}^{9}(N_{ij}+N_{ji})}, \tag{23}$$

$$F_{ST} = \frac{2\sum_{i=8}^{9} N_{ii}}{\sum_{i=8}^{9}\sum_{j=1}^{9}(N_{ij}+N_{ji})}. \tag{24}$$

5.1.2 PhysioNet Dataset

The publicly available training set of the PhysioNet 2017 Atrial Fibrillation Detection Challenge [38] has 8,528 recordings of single-lead ECGs with a time duration of 9-60s and a sampling rate of 300Hz. The dataset consists of four classes: 5,050 normal recordings, 738 atrial fibrillation recordings, 2,456 "other rhythms" recordings, and 284 noisy recordings, where the numbers are counted from the downloaded dataset.

5.1.3 CKB Dataset

The China Kadoorie Biobank (CKB) [46] is publicly available for bonafide researchers at http://www.ckbiobank.org/site/Data+Access. The standard 12-lead ECGs (10s duration, sampled at 500Hz) were recorded for 24,959 participants. After removing 113 participants with incomplete records, the ECG records collected from the remaining 24,906 participants were used to support this study.

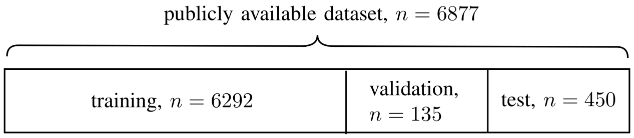
5.2 Experiment Configuration

All LCN models were trained using Adam with default hyperparameters ($\beta_1 = 0.9, \beta_2 = 0.999$) and the default learning rate of 0.001. The Hannun-Rajpurkar model, as a benchmarking approach, was trained using the authors' original implementation (https://github.com/awni/ecg) to ensure identical implementation. In brief, Hannun-Rajpurkar model used Adam [47] with a learning rate scheduler that decreases the learning rate after no improvement in the validation loss for two epochs. All hyperparameters were kept the same as described in the provided code [1]. All models were trained using early stopping (parameter "patience" = 8 epochs) with a maximum of 100 epochs for training [1]. All experiments were performed on Ubuntu 18.04, CPU with 32G RAM, single Nvidia GeForce GTX 1080 GPU, Python version 2.7.15, and Tensorflow version 1.8.0.

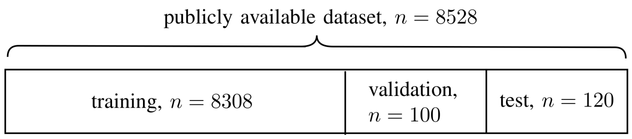
5.3 Experimental Validation on ECG Datasets

5.3.1 Validation on ICBEB Dataset

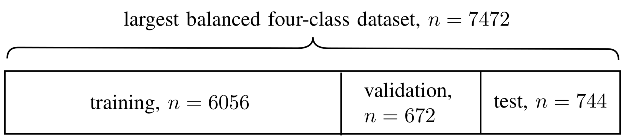
We divided the dataset into training, validation, and test sets as shown in Fig. 4a. We constructed balanced datasets by maintaining the same class distribution across all sets. Lacking access to the hidden test set, we randomly sampled 50 examples from each class in the publicly available training portion ($n = 6,877$) to build a balanced test set ($n = 450$), resulting in the same size and class distribution as the ICBEB Challenge. Similarly, we sampled another 15 examples per



(a) ICBEB.



(b) PhysioNet.



(c) CKB.

Figure 4: Training-Validation-Test Split of each dataset.

class to form a balanced validation set. We repeated the split and experiment five times. In each repeat, all models shared the same training, validation, and test sets.

The samples were weighted by the inverse of their class ratio in the training set. For example, if class i has n_i samples in the training set, then each sample of class i receives $\frac{\sum_i n_i}{n_i}$ weight during training. Since the pooling size is fixed for both the LCN and Hannun-Rajpurkar models during training, these models require the input signals to have the same number of data points. Ideally, the target length should be the maximum signal duration in the training set, i.e., 61s. However, due to memory constraints, we could only feed in signals with the duration of 37s. Therefore, the target length of signals for ICBEB is 37s. If the original signal is shorter than the target length, zeros are padded to the end of the signal. If the signal is longer than the target length, it is truncated at the end.

In each repeat, AutoNet identifies the "best" ReLU-LCN model and the "best" Leaky-LCN model separately. The hyperparameter n_f is calculated according to equations (17) and (18) with $n_f = 20$. The $n_{maxpool}$ is calculated as 9 according to equation (16) with $f_s = 500Hz$, $\tau = 1s$, $p = 2$. It took 1h 25min (5,095s) on average for the AutoNet to identify the best ReLU-LCN model, and 1h 55min (6,936s) to identify the best Leaky-LCN model. For ReLU-LCN, three out of five repeats converged at $n_{repeat} = 5$ with both skip connections and batch normalisation (Fig. 5a); one experiment converged at $n_{repeat} = 6$, with both skip connections and batch normalisation; and one experiment converged at $n_{repeat} = 4$, with both skip connections and batch normalisation (Table 3). For Leaky-LCN, four out of five repeats converged at $n_{repeat} = 5$, with both skip connections and batch normalisation, while the other repeat converged at $n_{repeat} = 7$, with both skip connections and batch normalisation.

Model architectures and training characteristics of ReLU-LCN, Leaky-LCN, and the Hannun-Rajpurkar model are shown in Table 1. The number of parametric layers represents the most frequently found architecture among the five experiments, the speed (s/epoch) and total epochs are the average values over the five experiments. The runtime is calculated by equation (25). The identified "best" architectures were identical for ReLU-LCN and Leaky-LCN, both have only 2.3% parameters compared to the Hannun-Rajpurkar model. Both ReLU-LCN and Leaky-LCN converged to deeper architectures compared to the Hannun-Rajpurkar model, supporting our hypothesis about the par-

Table 1: The architecture and training characteristics of ReLU-LCN, Leaky-LCN, and the Hannun-Rajpurkar models on ICBEB. conv: convolutional layer; BN: batch normalisation; TDS: time distributed softmax.

	ReLU-LCN	**Leaky-LCN**	**Hannun-Rajpurkar**
Train size	6,427	6,427	6,427
Test size	450	450	450
Batch size	32	32	32
Parametric layers	84 (41 conv, 42 BN, 1 TDS)	84 (41 conv, 42 BN, 1 TDS)	67 (33 conv, 33 BN, 1 TDS)
Parameters (%)*	239,596 (2.3)	239,596 (2.3)	10,473,322 (100)
Speed (s/epoch)	36	41	91
Total epoch	27	30	21
Runtime (s, %)*	955 (50.0)	1,248 (65.3)	1,911 (100)

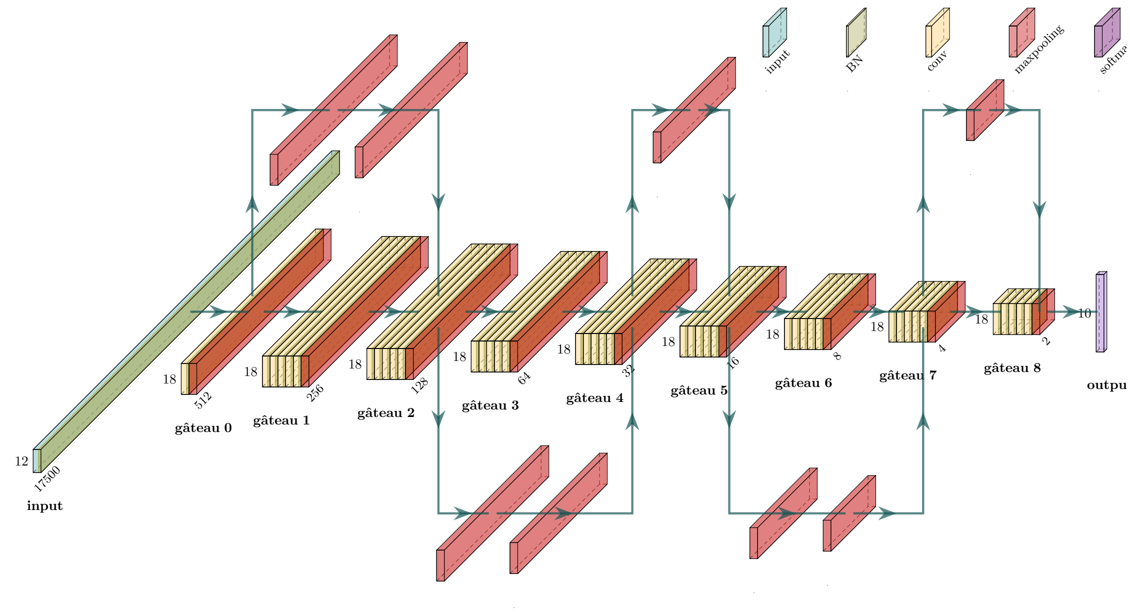
* % relative to the Hannun-Rajpurkar model.

Table 2: Mean and standard deviation (SD), mean ± SD, of the test F_1 scores from five experiments by ReLU-LCN, Leaky-LCN, and Hannun-Rajpurkar models on ICBEB. The highest mean F_1 score of each category is in bold font. No model averaging was performed.

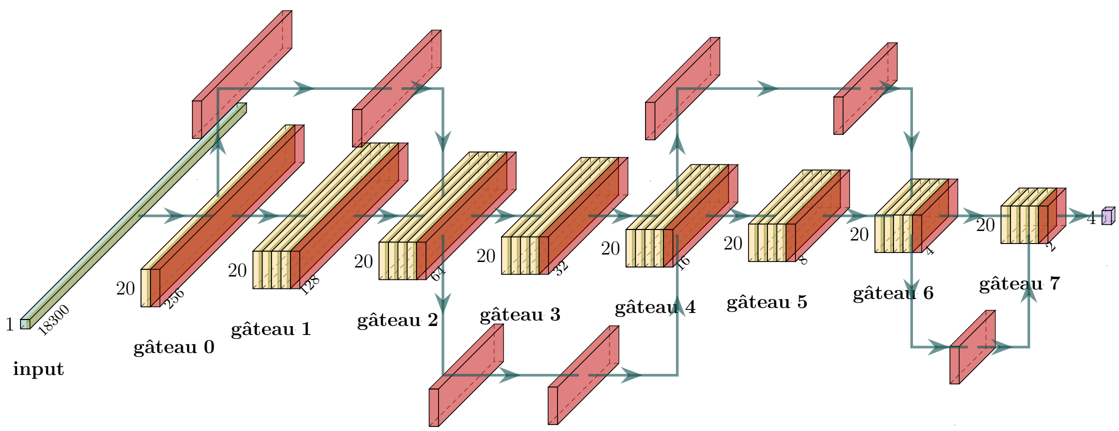
	Training size	**ReLU-LCN**	**Leaky-LCN**	**Hannun-Rajpurkar**
N	868	64.1±3.8	64.8±6.0	**69.8±4.4**
AF	1,048	84.2±3.3	**85.4±1.4**	84.7±3.7
I-AVB	654	84.2±1.9	85.2±3.1	**86.0±3.7**
LBBB	1,57	**89.1±1.7**	88.7±2.4	88.0±2.0
RBBB	1,645	76.5±3.4	**78.4±4.6**	76.0±4.1
PAC	506	64.8±12.6	**67.5±4.3**	61.4±9.7
PVC	622	81.4±4.7	**83.1±2.7**	80.1±5.6
STD	775	68.1±6.9	76.2±5.1	**78.9±4.7**
STE	152	68.1±3.9	**69.2±2.8**	58.3±7.7
9-class F_1		75.6±3.6	**77.6±2.0**	75.9±2.9
F_{AF}		84.2±3.3	**85.4±1.4**	84.7±3.7
F_{Block}		83.3±2.1	**84.1±2.1**	83.0±2.3
F_{PC}		72.0±9.3	**75.0±3.1**	70.7±7.1
F_{ST}		68.1±4.5	**72.5±3.0**	69.9±4.0

Table 3: The hyperparameters of the LCN models found on the five ICBEB experiments. "+" indicates "Yes". The most common architectures are in bold font.

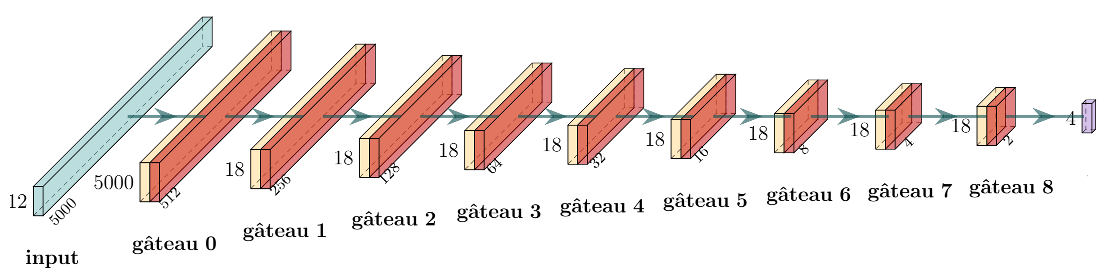
Repeat	**ReLU-LCN**			**Leaky-LCN**		
	n_{repeat}	$skip$	bn	n_{repeat}	$skip$	bn
1	**5**	**+**	**+**	7	+	+
2	6	+	+	**5**	**+**	**+**
3	4	+	+	**5**	**+**	**+**
4	**5**	**+**	**+**	**5**	**+**	**+**
5	**5**	**+**	**+**	**5**	**+**	**+**



(a) Auto-generated ReLU-LCN for ICBEB: n_{repeat} = 5, $n_{maxpool}$ = 9, meaning there are a total of 9 max-pooling layers, and there are five convolutional layers stacked between every two max-pooling layers. Batch normalisation is added after the input layer and after each convolutional layer. The after-convolution tensor is added to every 8 subsequent after-convolutional tensors, which are labelled in the figure. The output layer is a time-distributed 10-unit softmax layer, one unit for each of the nine classes and one unit to indicate noise/zero paddings.



(b) The most commonly auto-generated Leaky-LCN for PhysioNet: $n_{repeat} = 4$, $n_{maxpool} = 8$, $c = k = 20$. A batch normalisation layer (green) is added after the input layer and after every convolutional layer. A after-convolution tensor is added to every 7 subsequent after-convolution tensors.



(c) Auto-generated network for CKB: $n_{repeat} = 3$, $n_{maxpool} = 9$, $n_f = k = 18$. No batch normalisation nor skip connection was needed. The output is a 4-unit time distributed softmax layer.

Figure 5: Visualisation of the auto-generated LCNs on three datasets. The activation can be ReLU or leaky ReLU, which follows every convolutional layer, not shown in the figure to declutter the diagram. See Fig. 3 for magnified connection structure.

simony of LCN promoting deeper models.

$$runtime = \frac{1}{5}\sum_{i=1}^{5} total\ epoch \times speed \tag{25}$$

Both LCN models trained on each epoch faster than the Hannun-Rajpurkar model, although the latter converged after fewer epochs (Table 1). Both LCN models also have much shorter runtime compared to the Hannun-Rajpurkar model. Training speed depends on architecture, input signal length, and batch size. Longer signals and smaller batch sizes lead to slower training. Therefore, the runtime difference between the LCN models and the Hannun-Rajpurkar model is less dramatic than the parameter comparison. On average, Leaky-LCN requires more runtime than ReLU-LCN, as the Leaky-LCN tends to find deeper models.

Table 2 shows the testing accuracy F_1 scores for the three models. Leaky-LCN has the highest mean value in most ECG classes, while ReLU-LCN performs similarly to Hannun-Rajpurkar in most cases. For sub-abnormal groups and the 9-class F_1 score (used as the Challenge's evaluation criteria), Leaky-LCN consistently outperforms the other two models. Surprisingly, all three models achieved their best performance in the LBBB class, even though LBBB is the second smallest class in the training set. This is likely due to the fact that LBBB has clear clinical ECG diagnosis criteria. The model performances did not show a strong correlation with the training size. For example, STE has a similar number of training examples as LBBB but is poorly classified. This suggests that certain medical conditions, like STE, are inherently difficult for CNN-based architectures to classify from ECG, aligning with the clinical knowledge that some conditions lack definitive ECG characteristics.

5.3.2 Validation on PhysioNet Dataset

For the PhysioNet dataset, as shown in Fig. 4b, we randomly selected 30 samples (approximately 10% of the smallest class) from each class to build a balanced test set ($n = 120$), and another 25 samples (roughly 9% of the smallest class) from each class to build a balanced validation set, and the rest samples of the dataset were used for model training. The samples were weighted using the same procedure as described in section 5.3.1, and they were padded following the guidelines in Section 5.3.1.

AutoNet identifies the "best" ReLU-LCN model and the "best" Leaky-LCN model separately in each repeat. The hyperparameter n_f is calculated as $n_f = 20$ according to equations (17) and (18). The $n_{maxpool}$ is calculated as $n_{maxpool} = 8$ according to equation (16) with $f_s = 300Hz$, $\tau = 1s$, $p = 2$. It took 53 min (3203s) on average for the AutoNet to identify the best ReLU-LCN model, and 1h 30min (5413s) to identify the best Leaky-LCN model. For ReLU-LCN, two out of five repeats converged at $n_{repeat} = 2$ without skip connections and batch normalisation (Table 6); One experiment converged at $n_{repeat} = 2$, with only skip connections and without batch normalisation; One experiment converged at $n_{repeat} = 3$, with both skip connections and batch normalisation; and the other repeat converged at $n_{repeat} = 4$ with only skip connections and without batch normalisation. For Leaky-LCN, four out five repeats converged at $n_{repeat} = 4$, with both skip connections and batch normalisation (Fig. 5b), and the other repeat converged at $n_{repeat} = 5$, with only skip connections and without batch normalisation.

Table 4: The architecture and training characteristics of ReLU-LCN, Leaky-LCN, and the Hannun-Rajpurkar model on PhysioNet. conv: convolutional layer; BN: batch normalisation; TDS: time distributed softmax.

	ReLU-LCN	Leaky-LCN	Hannun-Rajpurkar
Training size	8,308	8,308	8,308
Test size	120	120	120
Batch size	32	32	32
Parametric layers	16 (15 Conv, 1 TDS)	60 (29 conv, 30 BN, 1 TDS)	67 (33 conv, 33 BN, 1 TDS)
Parameters (%)*	112,784 (1.1)	226,226 (2.2)	104,661,48 (100)
Speed (s/epoch)	20.6	43.2	121
Total epoch	30	28	21
Runtime (s,%)	611 (23.6)	1,207 (46.6)	2,589 (100)

* % relative to the Hannun-Rajpurkar model.

Table 5: Mean and standard deviation (SD), mean ± SD, of the test F_1 scores in five experiments by ReLU-LCN, Leaky-LCN, and Hannun-Rajpurkar models on PhysioNet. The highest F_1 score of each category is in bold font. No model averaging was performed.

	Training size	ReLU-LCN	Leaky-LCN	Hannun-Rajpurkar
AF	708	**88.8±2.8**	80.4±2.3	87.9±4.2
Normal	5,020	80.3±3.6	**86.4±4.3**	77.0±2.0
Other rhythms	2,426	72.3±7.7	**79.5±3.7**	74.6±3.8
Noise	254	**87.9±4.3**	72.4±4.6	74.7±6.1
F_{14}		82.3±3.1	**83.3±5.2**	78.5±3.3
F_{13}		**80.5±3.6**	79.5±1.5	79.8±2.6

Table 6: The hyperparameters of the LCN models found on the five PhysioNet experiments. "+" indicates "Yes", and "-" indicates "No". The most common architectures are in bold font.

Repeat	ReLU-LCN			Leaky-LCN		
	n_{repeat}	$skip$	bn	n_{repeat}	$skip$	bn
1	3	+	+	**4**	**+**	**+**
2	4	+	-	5	+	-
3	2	+	-	**4**	**+**	**+**
4	**2**	**-**	**-**	**4**	**+**	**+**
5	**2**	**-**	**-**	**4**	**+**	**+**

Model architectures and training characteristics of the three models are shown in Table 4. The LCN models have no more than 2.2% of the parameters than those of the Hannun-Rajpurkar model. Similar conclusions are drawn regarding runtime, total epochs, and training speed in the ICBEB and PhysioNet experiments, suggesting the consistent performance of the LCNs on different datasets. Table 5 shows the test F_1 scores of the three models. ReLU-LCN excels at identifying atrial fibrillation and noise, while Leaky-LCN outperforms other models in classifying normal and other rhythms. Notably, all three models show no bias towards larger classes, indicating the effectiveness of the sample weighting mechanism.

Table 7: The architecture and training characteristics of ReLU-LCN, Leaky-LCN, and the Hannun-Rajpurkar model on CKB. conv: convolutional layer; BN: batch normalisation; TDS: time distributed softmax.

	ReLU-LCN	Leaky-LCN	Hannun-Rajpurkar
Training size	6,728	6,728	6,728
Test size	744	744	744
Batch size	32	32	32
Parametric layers	10 (9 conv, 1 TDS)	10 (9 conv, 1 TDS)	67 (33 conv, 33BN, 1 TDS)
Parameters (%)*	50,782 (0.5)	50,7872 (0.5)	10,471,780 (100)
Speed (s/epoch)	4	5	34
Total epoch	24	20	13
Runtime (s, %)*	95 (21.5)	97 (22.0)	442 (100)

* % relative to the Hannun-Rajpurkar model.

Table 8: Mean and standard deviation (SD), mean ± SD, of the test F_1 scores on five experiments by ReLU-LCN, Leaky-LCN, and Hannun-Rajpurkar models on CKB. The highest F_1 score of each category is in bold font. No model averaging was performed.

	Training size	ReLU-LCN	Leaky-LCN	Hannun-Rajpurkar
Arrhythmia	1,681	**74.0±1.4**	71.7±3.7	63.7±10.1
Hypertrophy	1,681	**85.2±1.5**	82.5±1.0	75.2±16.8
Ischemia	1,681	72.4±2.6	**73.2±2.0**	66.9±2.2
Normal	1,681	**77.2±2.9**	75.6±2.7	69.5±3.3
4-class F_1		**77.2±1.6**	75.8±1.9	68.9±4.6

5.3.3 Validation on CKB Dataset

For the CKB dataset, we constructed a balanced set of normal, arrhythmia, ischemia, and hypertrophy classes by randomly sampling 1,868 (the size of the smallest class) recordings from each of the four classes. The resulting set was then stratified into training, validation, and test sets, respectively (Fig. 4c). The sampling and split were repeated five times to generate the training, validation, and test sets. In each repeat, the training, validation, and test sets were shared among all models. The procedure for sample weighting is described in 5.3.1, and all signals in the CKB dataset have the same duration and sampling rate (10s, 500Hz), thus there is no need for signal padding.

The hyperparameter n_f is calculated according to equations (17) and (18) with $m = 6,056$, thus $n_f = 18$. $n_{maxpool}$ is calculated as 9 according to equation (16) with $f_s = 500Hz$, $\tau = 1s, p = 2$. It took approximately 7 min (427s) on average for the AutoNet to identify the best ReLU-LCN model, and 11 min (693s) to identify the best Leaky-LCN model. For ReLU-LCN, all five repeats converged at $n_{repeat} = 1$ without skip connections nor batch normalisation (Fig. 5c); for Leaky-LCN, three out of five repeats converged at $n_{repeat} = 1$, without skip connections nor batch normalisation, while the other two repeats converged at $n_{repeat} = 2$, with only skip connections and without batch normalisation (Table 9).

Model architectures and training characteristics of the three models are shown in the Table 7. Both LCN models converged at nine convolutional layers without the need of batch normalisation, with only 0.5% parameters and much shorter runtime than the Hannun-Rajpurkar model. Table 8 shows the testing accuracy F_1 scores for the three models. LCN models outperformed the Hannun-Rajpurkar model in all categories, with 8-16% improvement in performance depending on the category and model. ReLU-LCN performed best in most cases, except ischemia, while the difference between ReLU-LCN and Leaky-LCN was not significant. As both training and test sets are balanced, the performance differences of the same model stems solely from the inherent characteristics of the medical condition. Arrhythmia and ischemia were more difficult than other classes for all three models, while hypertrophy was the easiest pattern to be identified. This agrees with the result in ICBEB (section 5.3.1) where LBBB was the best classified. This aligns with the finding in Section 5.3.1 that LBBB was the easiest pattern for identification.

Table 9: The hyperparameters of the LCN models found on the five CKB experiments. "+" indicates "Yes", and "-" indicates "No". The most common architectures are in bold font.

Repeat	ReLU-LCN			Leaky-LCN		
	n_{repeat}	$skip$	bn	n_{repeat}	$skip$	bn
1	**1**	**-**	**-**	2	+	-
2	**1**	**-**	**-**	**1**	**-**	**-**
3	**1**	**-**	**-**	2	+	-
4	**1**	**-**	**-**	**1**	**-**	**-**
5	**1**	**-**	**-**	**1**	**-**	**-**

5.4 Additional Validation on Non-ECG Datasets

Apart from validating our proposed AutoNet-LCN model on the aforementioned three ECG datasets, we conducted additional experiments on non-ECG datasets to further confirm the effectiveness of our model in broader classification tasks. It's worth noting the numerous of datasets available in the literature for benchmarking classification tasks. Our proposed AutoNet-LCN primarily aims to enhance the efficiency of developing an optimal model, particularly in handling large datasets. In this study, we refrain from considering small datasets (e.g., $m < 100$) for two reasons, firstly, manually tuning models on samll datasets is cost-prohibitive; and secondly, our AutoNet may compute model kernels with reduced values for smaller datasets, potentially restricting the model's capacity for feature learning.

We retrieved the following datasets to validate our developed model, (*i*) Spoken Arabic Digits (SAD) [1], and (*ii*) Face Detection (FD) [2]. The SAD dataset comprises 6,599 training samples and 2,199 testing samples for identifying 10 classes, each with the dimension of 93 × 13 for length and width. The FD dataset includes 5,890 training samples and 3,524 testing samples for 2 classes, with the dimension of 62 × 144 per sample. We either pad zeros or truncate signals to a length of 64 data points and use equations (17) and (18) to calculate the number of kernels for the AutoNet-LCN model. The kernel size is determined as $n_f = 18$, and the

1. http://www.timeseriesclassification.com/description.php?Dataset=SpokenArabicDigits
2. https://www.timeseriesclassification.com/description.php?Dataset=FaceDetection

Table 10: Mean and standard deviation (SD), mean ± SD, of the test F_1 scores on five experiments by ReLU-LCN and Leaky-LCN on SAD.

Class	Training size	ReLU-LCN	Leaky-LCN
0	660	96.6 ± 1.3	98.8 ± 0.6
1	659	98.2 ± 1.2	99.0 ± 0.3
2	660	99.8 ± 0.3	99.7 ± 0.3
3	660	98.6 ± 0.9	99.4 ± 0.3
4	660	98.8 ± 1.0	98.5 ± 0.6
5	660	99.6 ± 0.3	99.6 ± 0.5
6	660	99.3 ± 1.1	99.8 ± 0.3
7	660	98.8 ± 0.9	99.6 ± 0.2
8	660	98.4 ± 1.0	98.5 ± 0.5
9	660	99.9 ± 0.1	99.6 ± 0.3
10-class F_1	660	98.8 ± 0.3	99.3 ± 0.1

Table 11: Mean and standard deviation (SD), mean ± SD, of the test F_1 scores on five experiments by ReLU-LCN and Leaky-LCN on FD.

Class	Training size	ReLU-LCN	Leaky-LCN
Scramble	2,945	66.8±0.7	67.8±0.4
Face	2,945	66.7±0.8	67.7±0.5
2-class F_1		66.8±0.6	67.7±0.2

Table 12: The hyperparameters of the LCN models found on the five SAD experiments. "+" indicates "Yes". The most common architectures are in bold font.

Repeat	ReLU-LCN			Leaky-LCN		
	n_{repeat}	$skip$	bn	n_{repeat}	$skip$	bn
1	**14**	**+**	**+**	**9**	+	+
2	11	**+**	**+**	8	**+**	**+**
3	**14**	**+**	**+**	**9**	+	+
4	9	**+**	**+**	**14**	**+**	**+**
5	**14**	**+**	**+**	**14**	**+**	**+**

number of max-pooling layers is set to $n_{maxpool} = 5$ for both datasets. As illustrated in Figs. 6 and 7, we searched the optimal architectures for the two datasets using our proposed AutoNet algorithm.

Tables 10 and 11 show the testing accuracy F_1 scores for the generated AutoNet-LCN models on the SAD and FD datasets. The Leaky-LCN model has higher mean values of accuracies than the ReLU-LCN on the two datasets. However, both models have moderate performance on the FD experiments, highlighting the challenges of the classification task on the FD dataset. We show architectures of the generated models in Tables 12 and 13. Table 12 shows that both the ReLU-LCN and Leaky-LCN use skip connection and batch normalisation for model development. For ReLU-LCN, three out of five repeats converged at $n_{repeat} = 14$ on the SAD; for Leaky-LCN, two out of five repeats converge at $n_{repeat} = 14$ on the FD. In Table 13, both models mostly do not use skip connection and batch normalisation. For ReLU-LCN, four out of five repeats converged at $n_{repeat} = 14$; For Leaky-LCN, the models converged at around $n_{repeat} = 3$, suggesting the Leaky-LCN tends to learn deeper architectures than the ReLU-LCN.

We also compared our developed AutoNet-LCN model with different types of machine learning models, including (*i*) six classical machine learning models, i.e., the dynamic time warping (DTW) model [48], XGBoost [49], Rocket

Table 13: The hyperparameters of the LCN models found on the five FD experiments. "+" indicates "Yes", and "-" indicates "No". The most common architectures are in bold font.

Repeat	ReLU-LCN			Leaky-LCN		
	n_{repeat}	$skip$	bn	n_{repeat}	$skip$	bn
1	2	**-**	**-**	2	**-**	**-**
2	**1**	**-**	**-**	3	**-**	**-**
3	**1**	**-**	**-**	4	**-**	**-**
4	**1**	**-**	**-**	7	+	+
5	**1**	**-**	**-**	1	**-**	**-**

[50], long short-term memory (LSTM) network [51], LSTNet [52], and dilated CNN [53]; (*ii*) As Transformer [54] and its variants have been demonstrated as powerful machine learning models in recent years, we therefore compared our AutoNet-LCN model with six transformer-based models, including Transformer [54], Reformer [55], Informer [56], Pyraformer [57], TimesNet [58], and FEDformer models [59].

Table 14: Performance comparison between twelve different machine learning models and AutoNet-LCN on the SAD and FD datasets.

	Models	SAD Dataset (Accuracy)	FD Dataset (Accuracy)
Classical models	DTW [48]	96.3	52.9
	XGBoost [49]	69.6	63.3
	Rocket [50]	71.2	64.7
	LSTM [51]	31.9	57.7
	LSTNet [52]	100	65.7
	DilatedCNN [53]	95.6	52.8
Advanced models	Transformer [54]	98.4	67.3
	Pyraformer [57]	99.6	65.7
	FEDformer [59]	100	66.0
	Informer [56]	100	67.0
	TimesNet [58]	99.0	68.6
	Flowformer [60]	98.8	67.6
Our model	**AutoNet-LCN**	99.3	67.7

Table 15: Parameters comparison between different machine learning models and AutoNet-LCN on the SAD and FD datasets.

	Models	SAD Dataset (Parameters (%)*)	FD Dataset (Parameters (%)*)
Advanced models	Transformer [54]	522,250	469,378
	Pyraformer [57]	785,514	526,306
	FEDformer [59]	433,629	921,225
	Informer [56]	725,393	672,521
	TimesNet [58]	1,203,434	8,297,346
	Flowformer [60]	6,780,426	6,475,266
Our model	**AutoNet-LCN**	334,098 (4.93)	140,312 (1.69)

% relative to the model with the largest number of parameters.

We present the comparison results of classification performance obtained using different machine learning models in Table 14. The table indicates that our developed AutoNet-LCN model achieves average F_1 scores of 99.2% and 67.7% on the two datasets respectively, outperforming five out of six traditional machine learning models on SAD and demonstrating superior performance compared to all six models on FD. Notably, our AutoNet-LCN model achieves performance comparable to transformer-based models. For

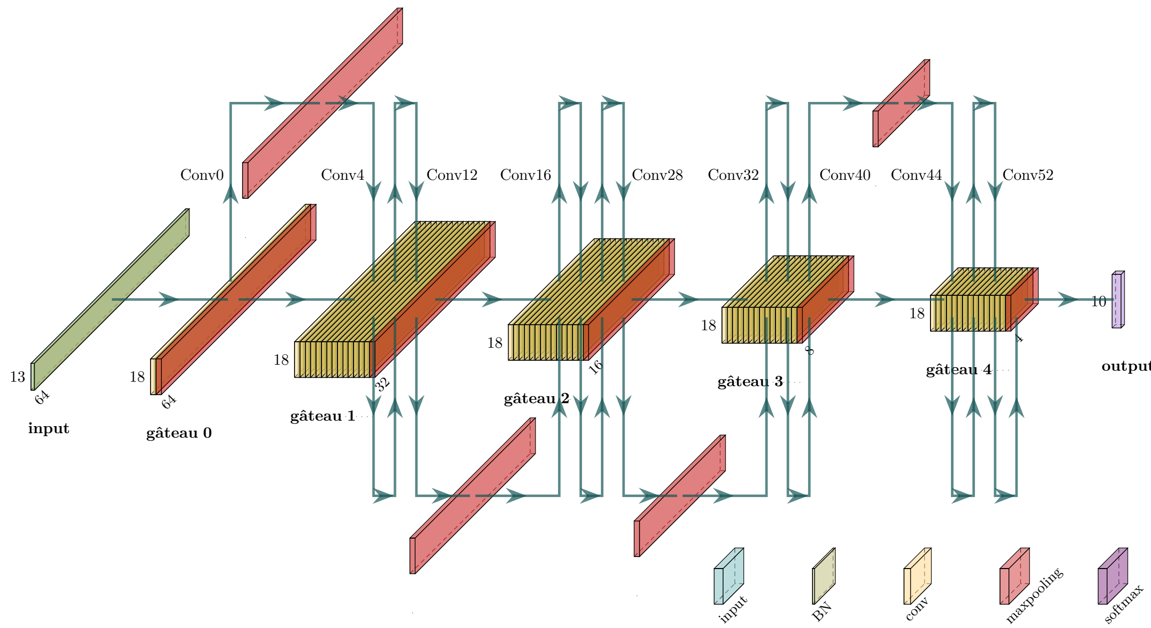


Figure 6: The auto-generated Leaky-LCN for SAD: $n_{repeat} = 14$, $n_{maxpool} = 5$, $c = k = 18$. The AutoNet algorithm searched the Leaky-LCN model with both max-pooling layer and no max-pooling layer for the skip connection.

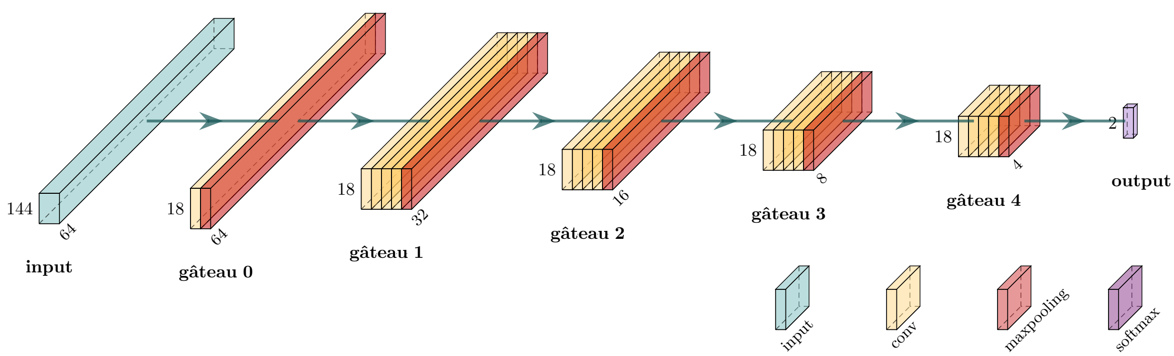


Figure 7: The auto-generated Leaky-LCN for FD: $n_{repeat} = 4$, $n_{maxpool} = 5$, $c = k = 18$. No batch normalisation nor skip connection was needed. The output is a 2-unit time distributed softmax layer.

instance, Table 15 demonstrates that our AutoNet efficiently identifies optimal models with significantly fewer parameters compared to transformer-based models. In particular, our AutoNet-LCN models have only 4.93% and 1.69% of the parameters of the largest transformer-based models on the two datasets respectively, underscoring the efficiency of our proposed AutoNet in discovering optimal models for classification tasks.

6 Discussion

One of the major contributions of this study is that our proposed LCN presents a novel paradigm to determine the hyperparameters of CNN. Central to the LCN theorem is the choice of n_f and f_w. In this study, the kernel size f_w is set to be equal to n_f. Theoretically, f_w should be independently optimised to maximise the total number of parameters in each layer, subject to $n_f(n_f k + 1) \leq m$. However, for long single-lead signals, such as those in PhysioNet, k would end up being unreasonably large (for example $f_w > 300$). Thus, we kept f_w to be the same as n_f. This also implicitly expresses our view that the parameters in the kernels and channel dimensions are not fundamentally different.

The resulting LCN in our study typically has less than 5% of parameters than the state-of-the-art models, indicating at least $O(n_{\boldsymbol{\theta}})$ saving in memory and computational complexity. The LCN may also make second-order algorithms feasible, as many second-order models need $O(n_{\boldsymbol{\theta}}^2)$ (conjugate gradient descent, BFGS) or $O(n_{\boldsymbol{\theta}}^3)$ (Newton method) complexity. If we optimise the parameters layer-by-layer, the computational complexity will be further reduced to be less than $O(m^2)$, where m is the number of training examples. Our future work will focus investigating the behaviour of convex optimisation in LCN networks.

This study uses multiple ECG datasets for experimental validation, each presenting unique challenges. The ICBEB dataset contains the most classes but has the fewest number of training examples per class. The PhysioNet dataset exhibits the highest ratio of noise and comprises only single-lead ECGs. The CKB dataset has ECGs with the shortest signal duration. When comparing performance on test sets across the three datasets, the lowest performance was observed with the CKB dataset. This suggests that the bottleneck of performance lies with the amount of information contained in each training example. It indicates that LCN can effectively utilise most of the training set. Furthermore, it is promising to observe that LCN performs well even with few training examples per class, which is often a limiting factor for deep learning models. Additionally, the simple sample weighting method effectively addresses class skewness, and the LCN models demonstrate minimal bias towards the larger classes.

It is worth noting the advantages of machine learning models with fewer parameters, such as reduced computational cost, and less model complexity, which in turn makes the model less sensitive to statistical fluctuation or noises in the input data. However, many neural networks in literature are over-parametrised, and it is hypothesised that the over-parametrised model would generalise better than the under-parametrised model [61], [62]. In fact, our generated AutoNet-LCN models are also over-parametrised. We note that this study proposed a new concept of layer-wise convex networks to develop deep learning models. We constrained the number of parameters in each layer of the network, rather than enforcing the whole neural networks to be over-parameterised as pointed out in [62]. However, the neural networks generated using our proposed AutoNet are still over-parameterised, which is consistent with the implications that over-parameterisation in neural networks can be beneficial [61], [62].

Our proposed LCN theorem was inspired by the "first principle" that each training example should contribute one "piece" of information to characterise one parameter in developing deep neural networks. Instead of formulating a black-box optimisation function as presented in many existing NAS frameworks [13], [14], [16], [21], [22], we leverage function approximation and information theory to introduce the LCN theorem. This theorem allows us to examine the relationships among the number of weights, biases, training data samples, activation functions, and the model architecture. Based on the LCN theorem, we developed a NAS framework (AutoNet-LCN) comprising two algorithms (Algorithms 1 and 2). This framework enables automatic search for optimal (or near-optimal) deep neural networks, rather than relying on the cost-expensive trial-and-error process or exhaustive search as used in many NAS frameworks.

We demonstrated the promising performance of our proposed NAS on three ECG datasets, and additionally evaluated its effectiveness on two non-ECG datasets. In all these experiments, our AutoNet-LCN model achieved superior or comparable performance to the state-of-the-art while having fewer model parameters. However, there is no universal approach to guide the design of deep learning models for arbitrary classification tasks, given the diversity in layer modules and optimization strategies. Furthermore, besides the experiments presented in this study, there are various types of datasets available for further validation [30], [31], [63]. This motivates us to further explore the potential of our proposed AutoNet-LCN model for broader tasks and applications in our next step research.

In this study, we focused on searching for optimal deep neural networks with CNN as model backbone. We acknowledge that transformer-based models have demonstrated promising performance across various tasks in the literature [54], [56], [57], [59], [64]. Although our proposed AutoNet-LCN cannot be directly applied for parameter optimization in these transformer-based models, the concept of our proposed AutoNet algorithm, with performance monitoring and adaptive building blocks, has the potential to improve the performance of these transformer-based models. One notable strength of our proposed AutoNet is its computational efficiency, with model training completing in less than 2 hours. In contrast, transformer-based models often have high computation costs. For instance, the HeartBEiT model, with 86 million parameters for processing 5 or 10-second ECGs, requires about 6 hours per epoch and around 2.5 months to train the model, which is impractical in resource-limited settings [65]. Our future research will focus on improving the performance of our AutoNet-LCN model for regression tasks and utilising NAS for optimizing other machine learning models, e.g., transformer-based models.

7 Conclusion

This work has a theoretical contribution to the neural architecture search through the introduction of a novel Layer-Wise Convex (LCN) Theorem. Applying our theory to the practical task of ECG classification, we proposed a new AutoNet algorithm for searching the optimal network. Validated on five diverse datasets, our AutoNet demonstrates its versatility by searching the optimal network architecture customised for each dataset. Remarkably, these generated architectures exhibit no more than 5% of the parameters found in state-of-the-art machine learning models. This research paves the way for efficient and effective methodologies on searching neural architectures for classification tasks.

References

[1] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," *Nature Medicine*, vol. 25, no. 1, p. 65, 2019.

[2] L. Lu, T. Zhu, A. H. Ribeiro, L. Clifton, E. Zhao, J. Zhou, A. L. P. Ribeiro, Y.-T. Zhang, and D. A. Clifton, "Decoding 2.3 million ECGs: Interpretable deep learning for advancing cardiovascular diagnosis and mortality risk stratification," *European Heart Journal - Digital Health*, 2024.

[3] A. H. Ribeiro, M. H. Ribeiro, G. M. Paixão, D. M. Oliveira, P. R. Gomes, J. A. Canazart, M. P. Ferreira, C. R. Andersson, P. W. Macfarlane, W. Meira Jr, *et al.*, "Automatic diagnosis of the 12-lead ECG using a deep neural network," *Nature Communications*, vol. 11, no. 1, p. 1760, 2020.

[4] R. Zhou, L. Lu, Z. Liu, T. Xiang, Z. Liang, D. A. Clifton, Y. Dong, and Y.-T. Zhang, "Semi-supervised learning for multi-label cardiovascular diseases prediction: A multi-dataset study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–17, 2023.

[5] W. Li, Y. M. Tang, K. M. Yu, and S. To, "SLC-GAN: An automated myocardial infarction detection model based on generative adversarial networks and convolutional neural networks with single-lead electrocardiogram synthesis," *Information Sciences*, vol. 589, pp. 738–750, 2022.

[6] Z. Liu, T. Zhu, L. Lu, Y.-t. Zhang, and D. A. Clifton, "Intelligent electrocardiogram acquisition via ubiquitous photoplethysmography monitoring," *IEEE Journal of Biomedical and Health Informatics*, 2023.

[7] T. Pokaprakarn, R. R. Kitzmiller, R. Moorman, D. E. Lake, A. K. Krishnamurthy, and M. R. Kosorok, "Sequence to sequence ECG cardiac rhythm classification using convolutional recurrent neural networks," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 2, pp. 572–580, 2021.

[8] G. Wang, C. Zhang, Y. Liu, H. Yang, D. Fu, H. Wang, and P. Zhang, "A global and updatable ECG beat classification system based on recurrent neural networks and active learning," *Information Sciences*, vol. 501, pp. 523–542, 2019.

[9] E. Eldele, M. Ragab, Z. Chen, M. Wu, C.-K. Kwoh, X. Li, and C. Guan, "Self-supervised contrastive representation learning for semi-supervised time-series classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[10] W. Zhang, L. Yang, S. Geng, and S. Hong, "Self-supervised time series representation learning via cross reconstruction transformer," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[11] H. Maennel, I. M. Alabdulmohsin, I. O. Tolstikhin, R. Baldock, O. Bousquet, S. Gelly, and D. Keysers, "What do neural networks learn when trained with random labels?," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19693–19704, 2020.

[12] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, 2021.

[13] Y. Li, M. Dong, Y. Wang, and C. Xu, "Neural architecture search via proxy validation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, pp. 7595–7610, 2023.

[14] Z. Chen, G. Qiu, P. Li, L. Zhu, X. Yang, and B. Sheng, "MNGNAS: Distilling adaptive combination of multiple searched networks for one-shot neural architecture search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–20, 2023.

[15] Z. Liu, H. Tang, S. Zhao, K. Shao, and S. Han, "PVNAS: 3D neural architecture search with point-voxel convolution," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 8552–8568, 2022.

[16] M. Zhang, H. Li, S. Pan, X. Chang, C. Zhou, Z. Ge, and S. Su, "One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2921–2935, 2021.

[17] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *International Conference on Learning Representations*, 2016.

[18] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2423–2432, 2018.

[19] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *International Conference on Learning Representations*, 2018.

[20] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*, pp. 2902–2911, PMLR, 2017.

[21] R. Hosseini and P. Xie, "Saliency-aware neural architecture search," *Advances in Neural Information Processing Systems*, vol. 35, pp. 14743–14757, 2022.

[22] Z. Lu, G. Sreekumar, E. Goodman, W. Banzhaf, K. Deb, and V. N. Boddeti, "Neural architecture transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2971–2989, 2021.

[23] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2018.

[24] Y. Shen, Y. Li, J. Zheng, W. Zhang, P. Yao, J. Li, S. Yang, J. Liu, and B. Cui, "ProxyBO: Accelerating neural architecture search via bayesian optimization with zero-cost proxies," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 9792–9801, 2023.

[25] C. Peng, A. Myronenko, A. Hatamizadeh, V. Nath, M. M. R. Siddiquee, Y. He, D. Xu, R. Chellappa, and D. Yang, "HyperSegNAS: Bridging one-shot neural architecture search with 3D medical image segmentation using hypernet," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20741–20751, 2022.

[26] X. Zhang, H. Xu, H. Mo, J. Tan, C. Yang, L. Wang, and W. Ren, "DCNAS: Densely connected neural architecture search for semantic image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13956–13967, 2021.

[27] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 82–92, 2019.

[28] L. Yao, H. Xu, W. Zhang, X. Liang, and Z. Li, "SM-NAS: Structural-to-modular neural architecture search for object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 12661–12668, 2020.

[29] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "DetNAS: Backbone search for object detection," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[30] G. Kong, C. Li, H. Peng, Z. Han, and H. Qiao, "EEG-based sleep stage classification via neural architecture search," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 31, pp. 1075–1085, 2023.

[31] S. Wang, H. Tang, B. Wang, and J. Mo, "A novel approach to detecting muscle fatigue based on sEMG by using neural architecture search framework," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[32] Z. Liu, H. Wang, Y. Gao, and S. Shi, "Automatic attention learning using neural architecture search for detection of cardiac abnormality in 12-lead ECG," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–12, 2021.

[33] J. Lv, Q. Ye, Y. Sun, J. Zhao, and J. Lv, "Heart-darts: classification of heartbeats using differentiable architecture search," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2021.

[34] H. Rakhshani, H. I. Fawaz, L. Idoumghar, G. Forestier, J. Lepagnot, J. Weber, M. Brévilliers, and P.-A. Muller, "Neural architecture search for time series classification," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2020.

[35] Y. Guan, Y. An, J. Xu, N. Liu, and J. Wang, "HA-ResNet: Residual neural network with hidden attention for ECG arrhythmia detection using two-dimensional signal," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2022.

[36] P. Bachtiger, C. F. Petri, F. E. Scott, S. R. Park, M. A. Kelshiker, H. K. Sahemey, B. Dumea, R. Alquero, P. S. Padam, I. R. Hatrick, *et al.*, "Point-of-care screening for heart failure with reduced ejection fraction using artificial intelligence during ECG-enabled stethoscope examination in London, UK: a prospective, observational, multicentre study," *The Lancet Digital Health*, vol. 4, no. 2, pp. e117–e125, 2022.

[37] E. M. Lima, A. H. Ribeiro, G. M. Paixão, M. H. Ribeiro, M. M. Pinto-Filho, P. R. Gomes, D. M. Oliveira, E. C. Sabino, B. B. Duncan, L. Giatti, *et al.*, "Deep neural network-estimated electrocardiographic age as a mortality predictor," *Nature Communications*, vol. 12, no. 1, p. 5117, 2021.

[38] G. D. Clifford, C. Liu, B. Moody, L.-w. H. Lehman, I. Silva, Q. Li, A. Johnson, and R. G. Mark, "AF classification from a short single lead ECG recording: The physionet computing in cardiology challenge 2017," *Proceedings of Computing in Cardiology*, vol. 44, p. 1, 2017.

[39] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[40] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[41] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[42] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, no. 6, pp. 861–867, 1993.

[43] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Networks*, vol. 3, no. 5, pp. 551–560, 1990.

[44] B. Hanin, "Which neural net architectures give rise to exploding and vanishing gradients?," *Advances in neural information processing systems*, vol. 31, 2018.

[45] F. Liu, C. Liu, L. Zhao, X. Zhang, X. Wu, X. Xu, Y. Liu, C. Ma, S. Wei, Z. He, *et al.*, "An open access database for evaluating the algorithms of electrocardiogram rhythm and morphology abnormality detection," *Journal of Medical Imaging and Health Informatics*, vol. 8, no. 7, pp. 1368–1373, 2018.

[46] Z. Chen, J. Chen, R. Collins, Y. Guo, R. Peto, F. Wu, and L. Li, "China Kadoorie Biobank of 0.5 million people: survey methods, baseline characteristics and long-term follow-up," *International Journal of Epidemiology*, vol. 40, no. 6, pp. 1652–1666, 2011.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, vol. 500, 2015.

[48] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pp. 359–370, 1994.

[49] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

[50] A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1454–1495, 2020.

[51] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[52] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st international ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018.

[53] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[55] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *International Conference on Learning Representations*, 2020.

[56] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 11106–11115, 2021.

[57] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *International Conference on Learning Representations*, 2021.

[58] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "TimesNet: Temporal 2D-variation modeling for general time series analysis," in *The Eleventh International Conference on Learning Representations*, 2022.

[59] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *International Conference on Machine Learning*, pp. 27268–27286, 2022.

[60] H. Wu, J. Wu, J. Xu, J. Wang, and M. Long, "Flowformer: Linearizing transformers with conservation flows," in *International Conference on Machine Learning*, pp. 24226–24242, PMLR, 2022.

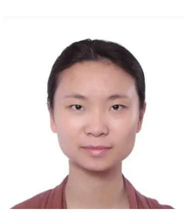
[61] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15849–15854, 2019.

[62] Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

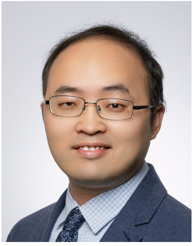
[63] J. Yan, L. Lu, D. Zhao, and G. Wang, "Diagnosis of bearing incipient faults using fuzzy logic based methodology," in *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 3, pp. 1229–1233, IEEE, 2010.

[64] F. Liu, T. Zhu, X. Wu, B. Yang, C. You, C. Wang, L. Lu, Z. Liu, Y. Zheng, X. Sun, *et al.*, "A medical multimodal large language model for future pandemics," *npj Digital Medicine*, vol. 6, no. 1, p. 226, 2023.

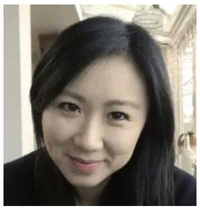
[65] A. Vaid, J. Jiang, A. Sawant, S. Lerakis, E. Argulian, Y. Ahuja, J. Lampert, A. Charney, H. Greenspan, J. Narula, *et al.*, "A foundational vision transformer improves diagnostic performance for electrocardiograms," *npj Digital Medicine*, vol. 6, no. 1, p. 108, 2023.



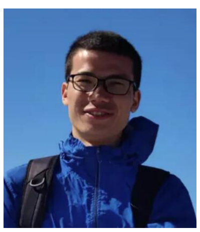
Yanting Shen studied PhD in machine learning at the Department of Engineering Science, University of Oxford, following B.Eng in biomedical engineering from Peking University. Following her graduation from Oxford, she held positions as a Machine Learning Scientist at AIG and Senior Data Scientist at Curve OS; Subsequently, she joined JPMorgan Chase as a Data Scientist Senior Associate, and now she is a Senior Machine Learning Engineer at Comcast NBCUniversal. Her interests include end-to-end machine learning research and engineering in healthcare, finance, and media, as well as LLM infrastructure for large-scale applications.



Lei Lu is a Lecturer in Health Data Science and AI at King's College London, and a Visiting Research Fellow at University of Oxford. He obtained his PhD from the Harbin Institute of Technology in China, complemented by two-year visiting research at the University of British Columbia in Canada. Lei had his postdoctoral research at the University of Melbourne in Australia, then he joined the Institute of Biomedical Engineering at University of Oxford as a Senior Research Associate. Lei's work focuses on clinical machine learning and computational informatics. He received the IET J.A. Lodge award in 2021, which presents to one early-career researcher annually with distinction.



Tingting Zhu received the B.Sc. degree in electrical engineering from the University of Malta, the M.Sc. degree in biomedical engineering from the University College London, and the D.Phil. degree in information engineering and biomedical engineering from the University of Oxford, in 2016. She is currently an Associate Professor at the Department of Engineering Science, University of Oxford. Her research interests include investigating the development of machine learning for understanding complex patient data, with a special emphasis on Bayesian inference, deep learning, and applications involving the developing world.



Xinshao Wang is a principal ML researcher of Terminal Industries, working on advancing and customising multi-modal large vision-language models and autoregressive generative models. He was a senior ML researcher of Zenith Ai and a visiting scholar of University of Oxford. He was a postdoctoral researcher at the Department of Engineering Science, University of Oxford after finishing his PhD at the Queens University of Belfast, UK. Xinshao Wang has been working on core deep learning techniques with applications to computer vision, natural language processing, disease prediction based on electronic health records, and protein engineering. Concretely, he has been working on the following research topics: (1) Deep metric learning: to learn discriminative and robust representations for diverse downstream tasks, e.g., object retrieval and clustering; (2) Robust deep learning: robust learning and inference under adverse conditions, e.g., noisy labels, missing labels (semi-supervised learning), out-of-distribution training examples, sample imbalance, etc; (3) Computer vision: video/set-based person re-identification; image/video classification/retrieval/clustering; object detection and recognition; image synthesis and generation; (4) AI healthcare: electrocardiogram classification; (5) ML-assisted gene and protein engineering; (6) advancing and customising multi-modal large vision-language models and autoregressive generative models.

Lei Clifton is currently the team leader and principal medical statistician of the Translational Epidemiology Unit, Nuffield Department of Population Health, Oxford University. She holds a PhD in Statistical Machine Learning from the University of Manchester, after completing her BSc and MSc degrees in Electrical Engineering at the Beijing Institute of Technology. Lei's research interest is at the intersection of machine learning and medical statistics, for both non-communicable and infectious diseases.

Zhengming Chen is a Professor qualified in medicine at Shanghai Medical University in 1983 (now Fudan University), and gained his DPhil in Epidemiology at the University of Oxford in 1993. He was appointed as Professor of Epidemiology by the University of Oxford in 2006. He is now the Director of the China Programs at the Oxford University's Clinical Trial Service Unit and Epidemiological Studies Unit (CTSU) and co-executive director of the China Oxford Centre for International Health Research. His main researches focus on the environmental and genetic causes of chronic disease, evidence-based medicine and evaluation of widely practicable treatments for chronic diseases (such as IHD, stroke and cancer) as well as efficient strategies for chronic disease control in developing countries. Over the past 20 years, he has led several large randomised trials in heart disease (eg, COMMIT/CCS-2), stroke (eg, CAST) and cancer and 3 cohort studies involving $>$750,000 individuals. Since 2003 he has been the lead principal investigator in the UK for the China Kadoorie Biobank (CKB) prospective study of 0.5 million adults. He leads a research team in Oxford which is responsible for study design and development of procedures and IT systems for the CKB, and for central data management, curation and detailed analyses. He is an honorary professor of Peking Union Medical College and Fudan University in China.

Robert Clarke is an Emeritus Professor of Epidemiology and Public Health Medicine at the Clinical Trial Service Unit and Epidemiological Studies Unit (CTSU), Honorary Consultant in Public Health Medicine and Scientific Director of the MSc in Global Health Science and Epidemiology. He qualified in clinical medicine in Ireland and worked for five years in internal medicine and four years in cardiology. After two years in clinical pharmacology at Vanderbilt University, Nashville, USA, he joined CTSU in 1991 and specialised in cardiovascular epidemiology. Over the last three decades in CTSU, his research work has focussed on understanding the causes of stroke and heart disease (ischaemic heart disease and heart failure) in prospective studies, including the China Kadoorie Biobank (CKB). His research work in CKB includes studies of the diagnostic accuracy, prognosis, treatment, risk prediction and evaluation of genetic and plasma biomarkers for stroke types and ischaemic heart disease and heart failure in CKB and related studies in the department and international consortia.

David Clifton is Royal Academy of Engineering Chair of Clinical Machine Learning, NIHR Research Professor, and leads the Computational Health Informatics Lab in the Department of Engineering Science at the University of Oxford. He is Official Fellow in AI & ML at Reuben College, Oxford, Research Fellow of the Royal Academy of Engineering, and Fellow of Fudan University, China.