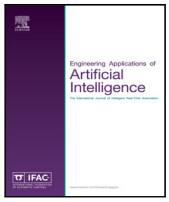
Engineering Applications of Artificial Intelligence 128 (2024) 107498

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Guided sampling-based evolutionary deep neural network for intelligent fault diagnosis

Arun K. Sharma [*,1], Nishchal K. Verma [2]

Department of Electrical Engineering, Indian Institute of Technology, Kanpur, India

ARTICLE INFO

Keywords:
Neural architecture search
Intelligent fault diagnosis
Deep neural network
Non-dominated sorting algorithm
Policy gradient

ABSTRACT

The selection of model architecture and hyperparameters has a significant impact on the diagnostic performance of most deep learning models. Because training and evaluating the various architectures of deep learning models is a time-consuming procedure, manual selection of model architecture becomes infeasible. Therefore, we have proposed a novel framework for evolutionary deep neural networks that uses a policy gradient to guide the evolution of the DNN architecture towards maximum diagnostic accuracy. We have formulated a policy gradient-based controller that generates an action to sample the new model architecture at every generation so that optimality is obtained quickly. The fitness of the best model obtained is used as a reward to update the policy parameters. Also, the best model obtained is transferred to the next generation for quick model evaluation in the NSGA-II evolutionary framework. Thus, the algorithm gets the benefits of fast non-dominated sorting as well as quick model evaluation. The effectiveness of the proposed framework has been validated on three datasets: the Air Compressor dataset, the Case Western Reserve University dataset, and the Paderborn University dataset.

1. Introduction

With the advancement in modern computational technology, machine learning-based intelligent fault diagnosis has become an integral part of almost all industrial sectors. Intelligent fault diagnosis refers to the preventive maintenance of rotating machines using machine learning-based data analysis and fault class detection (Nandi et al., 2005; Siddique et al., 2005; Yin et al., 2014; Chen et al., 2018b,a). Intelligent fault diagnosis, also termed as condition-based maintenance (CBM), has gained very much attention from interdisciplinary researchers. Different techniques such as statistical signal processing (He et al., 2010; Yu et al., 2017; Fan et al., 2018), fuzzy systems (Lei et al., 2008; Sharma et al., 2018), and deep neural networks (Su and Chong, 2007; Qi et al., 2017; Zhao et al., 2019) have been investigated in various literature for the development of intelligent diagnostic models. These intelligent systems aim to continuously monitor specific changes in machine signatures, including but not limited to vibration, acoustics, temperature, pressure, and provide timely notifications of anomalies or faults in various machine components (He et al., 2010; Yu et al., 2017). Diagnostic models are usually developed for laboratory machines. The developed model may fail to perform well for similar other machine deployed for industrial application due to different operating conditions. Running an industrial machine with a real-time load is an uneconomical task or sometimes impossible, especially in a faulty state, to acquire the diagnostic data. Therefore, the development of an intelligent fault diagnostic model for such machines faces the challenges of (i) training deep learning models with limited or unavailability of labeled data and (ii) selection of the best model architecture to avoid overfitting or underfitting due to limited availability of the training samples and to ensure reliable performance with the test samples.

To meet above challenges, several methodologies have been introduced such as cross-domain intelligent fault diagnosis with unavailability or limited availability of labeled data. For example, domain-adversarial training of neural networks (Ganin et al., 2016; Lu et al., 2017; Li et al., 2023b), cross-domain fault diagnosis under limited availability of labeled dataset (Li et al., 2019; Guo et al., 2019; Sharma and Verma, 2021). A quick learning mechanism solves the problem of training the deep learning model with limited availability of labeled target samples using net2net transformation followed by domain adaptation-based fine-tuning (Sharma and Verma, 2021). However, the diagnostic performance of all these methods is greatly affected by the selection of the deep neural network (DNN) model architecture.

* Corresponding author.
E-mail addresses: arnksh@iitk.ac.in (A.K. Sharma), nishchal@iitk.ac.in (N.K. Verma).
[1] Student Member, IEEE.
[2] Senior Member, IEEE.

https://doi.org/10.1016/j.engappai.2023.107498
Received 13 March 2023; Received in revised form 17 September 2023; Accepted 10 November 2023
Available online 27 November 2023

Therefore, our main objective is to investigate and develop an algorithm that can obtain a best-performing model for fault diagnosis with the dataset under variable operating conditions of industrial machines. There are an arbitrarily number of settings in every learning model that are orthogonal to the learning model itself (i.e., they live outside the model training process) but greatly affect the performance of the model. These parameters are known as hyper-parameters, which include learning rate, regularization parameters, and the model architecture design. Model architecture design or selection is one of the important aspects of hyper-parameter optimization, most popularly known as network architecture search (NAS) (He et al., 2021; Dudziak et al., 2020; Liu et al., 2022). NAS methods are mainly categorized as: (i) random search and grid search (Bergstra and Bengio, 2012), (ii) surrogate model-based optimization (Hutter et al., 2011; Fan and Wang, 2023; Wöhrle et al., 2023; Lu et al., 2022), (iii) reinforcement learning (Baker et al., 2016; Zoph and Le, 2016; Jaafra et al., 2019; Wang et al., 2020; Li et al., 2023a), (iv) genetic algorithm (Loghmanian et al., 2012; Wang et al., 2019; Sun et al., 2019, 2020, 2021b; Sharma and Verma, 2022; Qiu et al., 2023), (v) gradient descent (Liu et al., 2019, 2023), and (vi) hybrid algorithms (Yang et al., 2020). In the aforementioned methods, the biggest challenge for architecture search is the model evaluation because of the complex training mechanism for most of the DNN models (He et al., 2021). In genetic algorithm-based NAS methods (Loghmanian et al., 2012; Wang et al., 2019; Sun et al., 2019, 2020; Sharma and Verma, 2022), the hyper-parameters of the model to be optimized are encoded as an individual (chromosomes). A set of such individuals (population) is evaluated at each generation to evolve and find the best model. The overall efficacy of the evolution process depends on (i) the fitness evaluation strategy, (ii) the sorting, and (iii) the crossover and mutation strategies. The fitness evaluation involves the training and validation of the individuals (models) for the given dataset, which is a time-consuming process for deep neural networks. For example, regularized evolution of the image classifier (Real et al., 2019) with 450 K40 GPUs takes 3150 GPU days. Therefore, fault diagnosis with these NAS methods becomes infeasible as most industrial applications require a faster mechanism to quickly search for and train a suitable architecture of DNN.

Architecture optimization using fast non-dominated sorting genetic algorithm II (NSGA-II Deb et al., 2002) gets the benefits of a faster sorting method and therefore faster evolution (Lu et al., 2021). However, this method also requires the training and testing of individuals at each generation from scratch, and therefore becomes a time-consuming process for fault diagnosis applications. EvoN2N (Sharma and Verma, 2022) uses the concept of knowledge transfer for the evaluation of fitness in the framework based on NSGA-II. Quick fitness evaluation with fast NSGA-II makes the algorithm faster compared to state-of-the-art evolutionary NAS methods. This method uses crossover and mutation-based exploration and exploitation to find the best DNN architecture in the given search space. Similar to conventional genetic algorithms, this process requires a large number of generations to converge.

Therefore, this research work aims to propose a novel solution for (i) the development of the best-performing diagnostic model under limited training sample availability and (ii) reducing the computational cost with faster convergence of the NSGA-II-based model architecture search. We define fault classification accuracy and number of trainable parameters as the search objectives for model architecture optimization. To maximize the classification accuracy and minimize the number of total trainable parameters, We introduce a guided sampling-based evolution that makes convergence faster by exploiting the search space with the help of a reward-based controller instead of conventional crossover and mutation.

The key contribution to this work is guided sampling-based evolution of DNN architecture which includes following components:

(i) The **mean–variance-based mutation** to exploit the search space to obtain optimality.

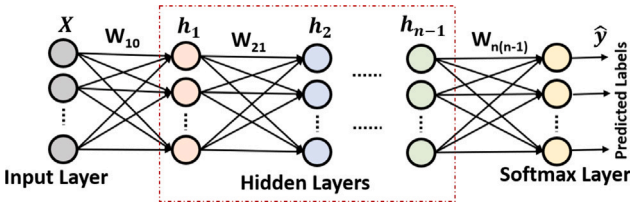


Fig. 1. SAE with softmax classifier (DNN model: Ψ).

(ii) The reward-based policy gradient controller to update mean and variance terms which guides the sampling of the DNN architecture to reach the optimality faster.

(iii) The quick model evaluation strategy is based on the knowledge transfer mechanism by transferring the knowledge of the best model obtained at every generation.

The remainder of the article is organized as follows. Section 2 briefly discusses the related works and theoretical background. Section 2.5 defines the objective problem. Section 3 explains the implementation details of the proposed framework of GS-EvoN2N. Section 4 discussed the effectiveness of the proposed framework for fault diagnosis under various machine load and operating conditions. And finally, Section 5 concludes the whole paper.

2. Related works and theoretical background

2.1. Deep Neural Network (DNN)

The deep neural network (DNN): a multi-layered neural network is the most popular technique for pattern recognition via non-linear feature transformation in multiple stages (Hinton and Salakhutdinov, 2006). From the training point of view, DNN can be considered as two parts: Stack of a given number of autoencoders (also called stacked autoencoder: SAE) (Bengio et al., 2007) and a classifier usually softmax classifier as output layer. First, greedy-layer unsupervised training is used to train each of the auto-encoder (AE) in the SAE. Then, the SAE stacked with the classifier at the end layer is fine-tuned using a labeled training dataset. The SAE with softmax classifier (DNN model Ψ) is depicted in Fig. 1.

2.2. Intelligent fault diagnosis

Recently, with the advent of advanced machine learning techniques and the availability of fast computational resources, the data-driven intelligent fault diagnosis method has gained much popularity, Nandi et al. (2005), Siddique et al. (2005), Chen et al. (2018b), Yin et al. (2014) and Li et al. (2023b,c). In these methods, various machine learning techniques are utilized to learn the specific signature of recorded signals like current, vibration, temperature, etc., and thereafter identify the existence of a machinery fault using the test samples. Neural network (NN) (Su and Chong, 2007), Support vector machine (SVM) (Widodo and Yang, 2007; Yan and Jia, 2018), and random forest (RF) classifier (Chen et al., 2018a) have been very effectively used for intelligent fault diagnosis and have been proved to be the baseline method for pattern recognition. But the diagnostic performances by these methods are reduced due to high sparsity and low-quality features in the dataset (Juan Jose et al., 2016). The intelligent fault diagnosis using deep learning methods has gained much attention due to its capability of multi-scale hierarchical feature transformation and large dimensional data handling, Qi et al. (2017), Zhao et al. (2019) and Guo et al. (2019). However, using a deep neural network for fault diagnosis faces a major challenge of training from scratch for every new operating condition of the machines. The recent trend of using deep transfer learning methods for domain adaptation has been very effective for

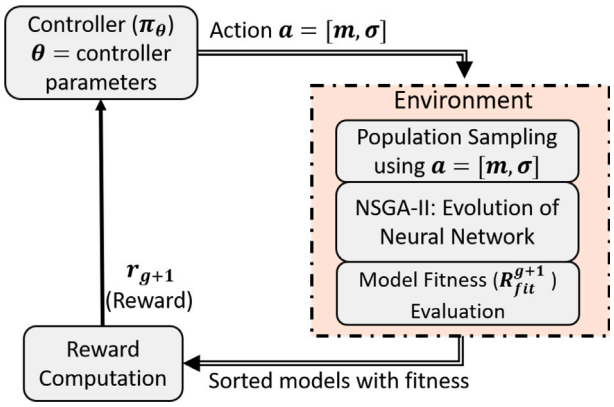


Fig. 2. Flow Diagram of guided sampling based NSGA-II.

fault diagnosis under changeable operating conditions (Pan et al., 2011; Long et al., 2014; Lu et al., 2017; Wen et al., 2019; Li et al., 2019; Wei et al., 2021; Sharma and Verma, 2021). Li et al. (2023b) introduces a deep adversarial network for remaining useful life (RUL) prediction. It utilizes global feature extraction and adversarial learning to ensure accurate RUL predictions, even when sensors are unreliable and prone to malfunction. However, the diagnosis performance by these methods is very much affected by the selection of the architecture of the deep neural network and the other hyper-parameters.

2.3. Neural Architecture Search (NAS)

The main objective of NAS methods is to obtain the optimal architecture in a given search space with the best model performance (He et al., 2021; Liu et al., 2022; Loni et al., 2018, 2020, 2022, 2019). There are three important aspects of the NAS methods: (i) formulation of the search space, (ii) architecture optimizer, and (iii) model evaluation. Formulation of search space defines the design format for the model architecture. It can be categorized into four groups: (i) cell-based (ii) entire-structured, (iii) morphism-based, and (iv) hierarchical search space. The most important aspect of NAS methods is the model evaluation as it is computationally very expensive to train each model during the search process and evaluate on the unseen dataset. To accelerate the evolution, various mechanism have been suggested for the model evaluation (He et al., 2021; Real et al., 2019; Zoph et al., 2018; Hundt et al., 2019; Hutter et al., 2011; Kandasamy et al., 2018; Bergstra et al., 2011; Pham et al., 2018). Kandasamy et al. (2018) suggested learning the curve extrapolation for the model performance evaluation instead to train and evaluate the actual architecture. Pham et al. (2018) proposed the parameter sharing method for faster training and evaluation of the model architecture.

Another important aspect of NAS methods is the architecture optimizer (AO). The objective automatic AO is to automatically guide the model architecture search in a direction to get the best suitable model for a given dataset. The AO methods adopted by various researchers can be categorized as (i) random search (RS) (ii) grid search (GS), (iii) surrogate model-based optimization (SMBO), (iv) gradient descent (GD), (v) reinforcement learning (RL), (vi) genetic algorithms (GA), and (vii) hybrid methods. In the RS method, the search optimizer tries different architecture randomly from the defined search space (Bergstra and Bengio, 2012), whereas, in GS, the search method uses a grid to sample and evaluate the model architecture (Hundt et al., 2019). SMBO methods use Basian optimization (Hutter et al., 2011; Kandasamy et al., 2018; Bergstra et al., 2011) or neural networks (Luo et al., 2018) as a surrogate model of the objective function to obtain the most promising solution (model architecture). Gradient descent-based method uses softmax function to find the optimal architecture over a continuous and differentiable search space (Liu et al., 2019). In RL-based NAS (Baker et al., 2016; Zoph and Le, 2016), a controller (usually, a recurrent neural network) generate an action to sample a new architecture. The observation (state) & the reward from the environment is used to update the controller policy to generate new architecture samples. Here, the training & validation process of the sampled neural network is treated as the environment that returns back the validation accuracy. GA-based NAS (Loghmanian et al., 2012; Wang et al., 2019; Sun et al., 2019, 2020, 2021a; Lu et al., 2021), use heuristic search to find the best performing architecture over a given search space. In these methods, heuristically sampled neural architectures are trained and evaluated using the convention neural training methods, and the performance metrics are used as fitness for evolution to obtain the optimal architecture. The main challenge of these methods is the fitness evaluation of the individual model. All of these methods of AO have their own merits and demerits. The hybridization of two of the above methods may give a significant improvement in the search efficiency, called the hybrid method of AO (Chen et al., 2019; Yang et al., 2020; Sun et al., 2019).

2.4. Policy gradient

Policy gradient (PG) is a tool to optimize the controller policy for reinforcement learning algorithm (Williams, May 1992; Sutton and Barto, 2018). The controller policy is the parameterized function that defines the learning agent's way to act on the environment to get maximum reward Fig. 2. The reward defines the good or bad effect of the action taken by the policy towards the fulfillment of the optimal objective. The policy may be defined as a deterministic or stochastic process. In deterministic policy, action is generated for every state of a deterministic environment, whereas stochastic policy generates a probability distribution of action for given states of the environment. Let the parameter vector be θ_t, then the parameterized function policy is represented as $\pi_{\theta_t}(a_t|s_t)$, where a_t and s_t represent action and state at given time t. Let the action a_t produces reward r_{t+1} from the environment, then the trajectory of state, action and reward can be represented as $\left((s_0, a_0, r_1), (s_1, a_1, r_2), \ldots .(s_t, a_t, r_{t+1})\right)$. The policy parameter θ_t can be updated using policy gradient as

$$\theta_{t+1} = \theta_t + \eta_t \nabla_{\theta_t} \mathcal{J}(\theta_t) \tag{1}$$

where η_t denotes the learning rate at time t, usually a constant real number. $\nabla_{\theta_t} \mathcal{J}(\theta_t)$ denotes the policy gradient and can be calculated using expected of cumulative reward U_t over the time t as follows.

$$\nabla_{\theta_t} \mathcal{J}(\theta_t) = \nabla_{\theta_t} E\left[U_t\right] = \nabla_{\theta_t} \int_t \pi(\tau) r(\tau) d(\tau) \tag{2}$$

$$= E\left[r(\tau).\nabla_{\theta_t} \log \pi_{\theta_t}(\tau)\right] \tag{3}$$

$$\nabla_{\theta_t} \mathcal{J}(\theta_t) = \frac{1}{N} \sum_{k=1}^{N} r(k) \left(\sum_{t=1}^{T} \nabla_{\theta_t} \log \pi_{\theta_t}\left(a_t | a_{t-1:1}; \theta_t\right) \right) \tag{4}$$

2.5. Problem statement

Let the training dataset, validation dataset, and test dataset are $\mathcal{D}^{tr} = \left(\mathbf{X}^{tr}, \mathbf{y}^{tr}\right)$, $\mathcal{D}^{val} = \left(\mathbf{X}^{val}, \mathbf{y}^{val}\right)$, and $\mathcal{D}^{te} = \left(\mathbf{X}^{te}, \mathbf{y}^{te}\right)$, respectively where $\mathbf{X} \in \mathfrak{R}^{(n_s \times n_f)}$ be the input data with n_s samples & n_f features and $\mathbf{y} \in \mathfrak{R}^{n_s}$ be the corresponding output label. The objective of optimal DNN architecture search for fault classification is mathematically be formulated as

$$\Psi^{\dagger} = \mathcal{H}\left(P,\ \mathcal{D}^{tr},\ \mathcal{D}^{val}\right) \tag{5}$$

$$\hat{y}^{te} = \mathcal{F}\left(\Psi^{\dagger},\ X^{te}\right) \tag{6}$$

where $\mathcal{H}(.)$ denotes the optimization function to get the best model $\Psi^{\dagger}$ with optimal parameters for the training dataset and $\mathcal{F}(.)$ is the feed-forward DNN function which predicts the fault class $\hat{\mathbf{y}}^{te}$ for the test data $\mathbf{X}^{te}$.

Algorithm 1 GS-EvoN2N: The Main Framework

Input: $\mathcal{D}^{tr}$ & $\mathcal{D}^{val}$ = training & validation datasets, n_R & h_R = Maximum depth & width of the DNN respectively.
Output: $\Psi^{\dagger}$ = best model after the termination or last generation of the evolution.

$t \longleftarrow 0$ //Set generation count (t) = 0;
$[m,\ \sigma] \longleftarrow$ Compute mean and variance from allowable range for depth (n_R) and width (h_R) of the DNN.
$P_0 \longleftarrow$ **GuidedPop**(m, σ, N_p) //Generate N_p number of populations using **Algorithm** 2.
$\Psi_0 \longleftarrow$ Initialize weight matrices of the first model ($P_0\{1\}$) by small random numbers.
$\Lambda, \Psi_1^{\dagger} \longleftarrow$ **FitnessEval**$(P_0, \mathcal{D}^{tr}, \mathcal{D}^{val}, \Psi_0)$ //Evaluate fitness of all individuals in P_0 using the **Algorithm** 3.
$\mathcal{R} \longleftarrow NonDominatedSorting(\Lambda_1)$ //Assign rank using non-dominated sorting (Deb et al., 2002).
$P_1 \longleftarrow SelectParents(P_0,\ \mathcal{R})$ //Select parents by binary tournament selection, Deb et al. (2002).
$\Lambda_s\{1\} \longleftarrow \Lambda$ //Store the fitness History.
$[m,\ \sigma] \longleftarrow$ **UpdateMeanVar**$(P_1,\ \Lambda_s)$ //Update mean and variance term using the **Algorithm** 4.
$Q_1 \longleftarrow$ **CrossoverMutation**$(P_1, m, \sigma))$ //Apply crossover and mutation on P using **Algorithm** 5.
$t \longleftarrow t+1$ //Update the generation count.
while *termination condition is false* **do**
$S_t \longleftarrow (P_t \cup Q_t)$ //Combine the parent population (P_t) & the child population (Q_t).
$\Lambda, \Psi_{t+1}^{\dagger} \longleftarrow$ **FitnessEval**$(S_t, \mathcal{D}^{tr}, \mathcal{D}^{val}, \Psi_t^{\dagger})$ //Evaluate fitness of all individuals in S_t using the **Algorithm** 3.
$\mathcal{R} \longleftarrow NonDominatedSorting(\Lambda)$ //Assign rank by non-dominated sorting of fitness Λ, Deb et al. (2002).
$\mathcal{K} \longleftarrow CrowdingDistances(S_t, \mathcal{R}, \Lambda)$ //Find crowding distances of individuals in population set S_t, Deb et al. (2002).
$P_{t+1} \longleftarrow SelectParentsByRankDist(S_t, \mathcal{K}, \Lambda)$ //Select parents by crowding distance and rank, Deb et al. (2002).
$\Lambda_s\{t+1\} \longleftarrow \Lambda$ //Store the fitness History.
$[m,\ \sigma] \longleftarrow$ **UpdateMeanVar**$(S_t,\ \Lambda_s)$ //Update mean and variance term using the **Algorithm** 4.
$Q_{t+1} \longleftarrow$ **CrossoverMutation**(P_{t+1}, m, σ) //Apply crossover and mutation on P using **Algorithm** 5.
  if Termination condition is true **then**
    Exit
  else
    $t \longleftarrow t+1$ //Update the generation counter.
  end if
end while
Return: **Best Model** : $\Psi^{\dagger} \longleftarrow \Psi_{t+1}^{\dagger}$ //Best model of the last generation.

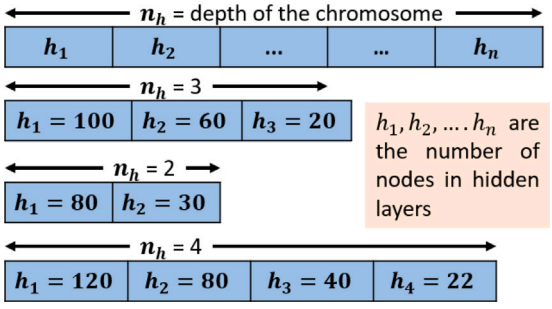


Fig. 3. Variable-length gene encoding strategy.

3. Proposed framework

In this section, the proposed framework of guided sampling-based evolutionary DNN (GS-EvoN2N) is described in detail. The Fig. 2 shows the schematic of the workflow of GS-EvoN2N. In the figure, DNN architecture optimization in the NSGA-II framework constitute the environment. The fitness of the best model is termed as the reward. The sorted fitness of all the individuals in the population is treated as the state of the controller. The controller policy π_{θ_t} generates an action $a_t = [m_t, \sigma_t]$, where m_t and σ_t be the mean and variance for the sampling of DNN architecture at generation t. Given the training dataset $\mathcal{D}^{tr}$ and the validation dataset $\mathcal{D}^{val}$, the algorithmic steps for the GS-EvoN2N is presented in Algorithm 1. Our contributions are highlighted in Algorithm 1 and are further discussed in the following sections.

3.1. Population sampling using mean and variance

AO includes depth and width variation in a defined search space. The real-coded gene encoding strategy is adopted to encode the depth as the number of genes (length of a chromosome) and the number of nodes in a hidden layer as the value of a gene as shown in Fig. 3. Let n_R & h_R be the maximum depth and width of the DNN, then the search space is defined as $[1\ n_R]$ & $[1\ h_R]$ for depth and the width variations. The mean and variance $m = [m_1, m_2]$ & $\sigma = [\sigma_1, \sigma_2]$ are initialized as $m_1 = (1+n_R)/2$, $m_2 = (1+h_R)/2$ and $\sigma_1 = (n_R-1)/2$, $\sigma_2 = (h_R-1)/2$.

Algorithm 2 GuidedPop: Population Sampling

Input: N = Population size, $m = [m_1, m_2]$ = mean & $\sigma = [\sigma_1, \sigma_2]$ = variance.
Output: P = Population with N chromosomes.

$H \longleftarrow$ generate N Gaussian numbers with m_1 and σ_1.
for p = 1 : N **do**
$h \longleftarrow H(p)$: depth of p^{th} chromosome
$tmp \longleftarrow$ generate h Gaussian numbers with m_2 and σ_2.
$P\{p\} \longleftarrow$ convert all numbers in tmp to nearest integers.
end for
Return P

3.2. Fitness evaluation

Fast model evaluation is the most important requirement for NAS, especially when the evolutionary algorithm is used as an AO strategy. If the best model at a generation is transferred for initialization of the DNN weight matrices in the next generation, it makes the training and evaluation of the models faster. The quick learning mechanism suggested in Sharma and Verma (2021) is adopted for fitness evaluation, as shown in Fig. 4. For the first generation, DNN models are randomly initialized and trained using the limited-Broyden–Fletcher–Goldfarb–Shanno (LBFGS) (Nocedal and Wright, 2006) algorithm. For next generation and later, the best model obtained is transformed (Fig. 4) to initialize the models followed by fine-tuning with the LBFGS algorithm for a few iterations only. If a model Ψ^t at generation tth

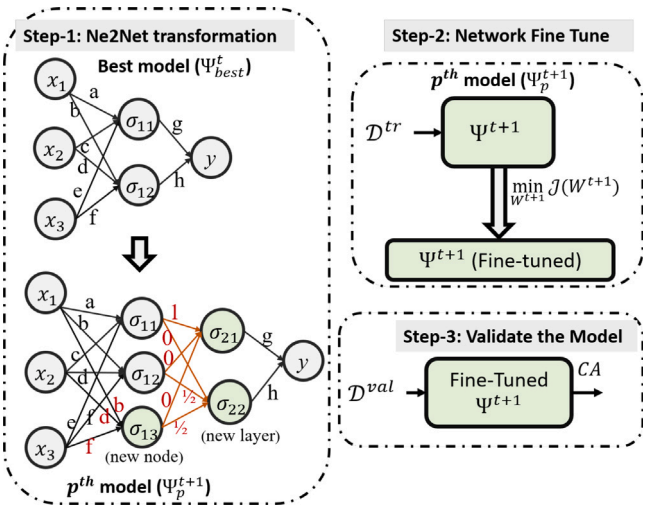


Fig. 4. Fitness evaluation strategy.

has weight matrix W^t, the classification loss $\mathcal{J}$ for a C problem can be defined in term of $[w, b] \in W^t$ as

$$\mathcal{J}(W^t) = \frac{1}{n^s}\left[\sum_{k=1}^{n^s}\sum_{i=1}^{C} I[y_k = c_i] \log \frac{e^{(w_i^T f(x_p)+b_i)}}{\sum_{i=1}^{C} e^{(w_i^T f(x_p)+b_i)}}\right] \tag{7}$$

where, $f(x) = \Phi(wx+b)$ is the h-level features representation of DNN, y_k be the output label of the kth data sample and c_i denotes the ith class. The classification accuracy (CA) of fine-tuned model for the validation data is returned as the fitness of that model.

Algorithm 3 FitnessEval: Fitness Evaluation

Input: P = Population with population size N_p,
$(\mathcal{D}^{tr}, \mathcal{D}^{val})$ = Training & validation data.
Output: Λ = Fitness matrix and $\Psi^\dagger$ = Best model.

$t \longleftarrow$ current generation
say $\Psi_t^\dagger \longleftarrow \Psi^\dagger$ // best model at current generation.
for $p = 1 : N_p$ **do**
  $\Psi_t^p \longleftarrow$ N2N($\Psi_t^\dagger$) // Transform $\Psi_t^\dagger$ to $\Psi_t^p \in P$ using N2N transformation as depicted in step-1 of Fig. 4.
  Fine-tune the model (Ψ_t^p) on $\mathcal{D}^{tr}$ to minimize Eq. (7).
  $\Lambda(p) \longleftarrow$ Find CA of Ψ_t^p on dataset $\mathcal{D}^{val}$.
end for
$\Psi_t^\dagger \longleftarrow$ Best model // Find the model with maximum CA and minimum number of model parameters.
Return Λ, $\Psi_t^\dagger$

3.3. Update mean and variance using PG

Mean (m) and variance (σ) terms are used for sampling of a new population as illustrated in Section 3.1. Here, we design a PG-based update laws for $m \,\&\, \sigma$ such that the best fitness ($\max(\Lambda)$) is maximized. At any generation t, $\max(\Lambda)$ is termed as reward, $a_t = [m, \sigma]$ is termed as action, and weighted average of fitness (Λ) is termed as state of the policy. Thus, the policy generates $[m^T, \sigma^T]$ to guide the evolution for faster and better convergence. For the design simplification, let us assume that the action is generated by a deterministic policy as

$$a_t = f(\theta_t) = \frac{1}{1+e^{-\theta_t}} \tag{8}$$

where, policy parameter θ_t is selected such that it controls $m \in \Re^2$ (mean of depth and width of DNN) and $\sigma \in \Re^2$ (variance for depth and width of DNN). The parameter θ is updated by the policy gradient in (1) calculated using gradient of expected total reward U_t derived in (4). The total cumulative reward is calculated using fitness matrix Λ_t at generation t as in (9).

$$U_t = \sum_{i=1}^{t} \frac{\max(\Lambda_i) - \max(\Lambda_{i-1})}{\max(\Lambda_{i-1})} \tag{9}$$

The algorithmic steps for the implementation of policy gradient based update of $a_t = [m, \sigma]$ at generation t is summarized in Algorithm 4.

Algorithm 4 UpdateMeanVar: Update m & σ using PG

Input: P_t = Current population, Λ = Fitness matrix,
$a_{t-1} = [m, \sigma]$ = Initial mean & variance term.
Output: $a_t = [m, \sigma]$ = Updated mean & variance.

N = Number of models in P_t, α = Learning rate
$\bar{n} \longleftarrow$ Compute average depth of models in P_t
$\Omega = [\omega_p]_{p=1}^N$ //Generate a set of weights ω_p such that $\sum_{p=1}^N \omega_p = 0$ and $\omega_1 > \omega_2 > \ldots. > \omega_N$
$\Lambda_t^{sorted}, idx = \text{sort}(\Lambda_t, \text{'descending'})$
$P_t \longleftarrow$ Sort P_t according to idx.
n_p = no. of hidden layers (depth) in p^{th} model.
$\delta_p = \max(H_p) - \min(H_p)$ //H_p = set of nodes in hidden layers of p^{th} model
if $t \leq 1$ **then**
  $m = \sum_{j=1}^N \left[n_p\omega_p \quad \frac{1}{n_p}\sum_{k=1}^{n_p} h_{kp}\omega_p\right]$ //$h_{kp} \in H_p$.
  $\sigma = \sum_{j=1}^N \left[(\bar{n} - n_p)\omega_p \quad \delta_p\omega_p/2\right]$
else
  $\theta_{t-1} = \ln\left[a_{t-1}/(1 - a_{t-1})\right]$
  $U_t \longleftarrow$ Compute cumulative reward using (9).
  $s_m = \sum_{j=1}^N \left[n_p\omega_p \quad \frac{1}{n_p}\sum_{k=1}^{n_p} h_{kp}\omega_p\right]$
  $s_\sigma = \sum_{j=1}^N \left[(\bar{n} - n_p)\omega_p \quad \delta_p\omega_p/2\right]$
  $s_t = [s_m^T \; s_\sigma^T]$
  $\theta_t \longleftarrow \theta_{t-1} + \alpha * \frac{1}{N}\sum_{k=1}^N U_t E[\nabla \log \pi_\theta(s_t|a_{t-1}; \theta)]$
  $a_t = 1/\left(1 + e^{-\theta_t}\right)$
end if
Return a_t

3.4. Crossover and mutation

For the optimal search of the network architecture, a combination of exploration and exploitation strategies is adopted. The guided sampling-based generation of new population exploits the search space to force the evolution towards maximum accuracy. To avoid local convergence, N number of individuals are sampled using m and σ based on Gaussian distribution, and also N number of parent populations are selected using crowding distance and rank from the current generation. After that, the two populations are merged to create a double-sized mating pool. Now, the two-step crossover operator introduced in Sharma and Verma (2022) is applied. The two steps are (i) single point depth crossover (SPDC) for depth variation and (ii) common depth simulated binary crossover (CDSBC) for gene value (width) alteration. The two-step crossover method is depicted in Fig. 5. The whole process of offspring generation is provided in Algorithm 5:

4. Experimental results and discussion

The efficacy of the proposed framework of GS-EvoN2N is demonstrated on fault diagnosis dataset under different operating conditions taken from (i) Air compressor fault data (Verma et al., 2016), (ii) Paderborn University (PBU) bearing fault data (Lessmeier et al., 2016), and (iii) CWRU bearing fault data (Smith and Randall, 2015).

Algorithm 5 Offspring Generation: Crossover and Mutation

Input: P = Parent population, p_c = Crossover probability, (m, σ) = mean & variance term for sampling.
Output: Q = Offspring population.

$Q \longleftarrow$ **GuidedPop**(m, σ, N, n_R, h_R) //Generate N offspring populations using Algorithm 2.
I_c = generate random indices of $p_c * 100\%$ members from P.
for $i \in I_c$ **do**
  Select $P_1 = P\{i\}$ & $P_2 = Q\{i\}$.
  Find lengths $(n_1,\ n_2)$ of $P_1,\ P_2$
  Set a point $h < \min(n_1, n_2)$ on $P_1,\ P_2$.
  $C_1, C_2 \longleftarrow$ **SPDC** of $P_1,\ P_2$ at point h (as depicted in **step-1 of** Fig. 5)
  $\bar{C}_1, \bar{C}_2 \longleftarrow$ **CDSBC** for genes of the common depth portion of C_1, C_2 as depicted in **step-2 of** Fig. 5.
  Replace $Q\{i\}$ by $\bar{C}_1$ or $\bar{C}_2$.
end for
Return Q

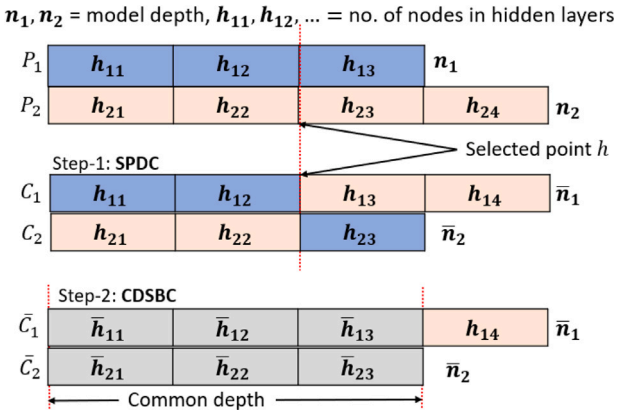


Fig. 5. Two-steps crossover for chromosomes with different length.

4.1. Experimental setup

4.1.1. Air compressor fault data (Verma et al., 2016)

The air compressor data contains acoustic signal recorded on single stage reciprocating type air compressor driven by an 5 hp induction motor installed at the workshop, EE Department, IIT Kanpur. Data were recorded in eight different cases: healthy and seven different faulty states of the air compressor valve. Therefore, the dataset has 8 classes: (i) Healthy (**H**), (ii) Leakage Inlet Valve (**LIV**), (iii) Leakage Outlet Valve (**LOV**), (iv) Non-Return Valve (**NRV**), (v) Piston Ring (**PR**), (vi) Flywheel (**F**), (vii) Rider Belt (**RB**), and (viii) Bearing (**B**). For each class, 225 measurements were taken with 50k samples in each measurement.

4.1.2. PBU bearing fault data (Lessmeier et al., 2016)

PBU bearing fault data is the collection of time-series signals recorded on electrical machine operating under wide variation of shaft load and rotational speed. The four Load and speed settings (LS) are **LS1: N09_M07_F10** (speed = 900 rpm, torque = 0.7 N m & radial force = 1000 N), **LS2: N15_M01_F10** (speed = 1500 rpm, torque = 0.1 N m & radial force = 1000 N), **LS3: N15_M07_F04** (speed = 1500 rpm, torque = 0.7 N m & radial force = 400 N), and **LS4: N15_M07_F1gr60** (speed = 1500 rpm, torque = 0.7 N m & radial force = 1000 N). Total of 32 experimentation with 6 healthy, 12 artificially damaged, and 14 damaged by long run accelerated tests were conducted to record current, vibration signal, radial forces, torque, and bearing temperature. The recorded signals contains two types of faults: inner race **(IR)** fault and outer race (**OR**) fault.

4.1.3. CWRU bearing fault data (Smith and Randall, 2015)

The CWRU bearing fault data provided by Case Western Reserve University (CWRU) has been a widely used benchmark dataset for bearing fault diagnosis. It contains vibration signals recorded at the drive end (DE) and the fan end (FE) of the artificially seeded bearing with inner race fault (**IR**), outer race (**OR**), and rolling element ball (**B**) faults of variable fault diameters (F.D.) (from 0.007 to 0.028 in.). The bearing test rig setup details can be found in Smith and Randall (2015).

4.2. Segmentation and data processing

The recorded time-series signals contain a huge number of samples which is not suitable for training the DNN. To make the dimension of the time-series signals compatible with the DNN, we adopt a segmentation rule with the segment length of approximately 1/4th of data points recorded per revolution. Here, we have selected segmentation lengths of 100, 200, & 400 for the CWRU dataset, Air compressor dataset, and PBU dataset respectively. Also, the time-series signals are usually unstructured and not to the scale. Therefore, we have applied the min–max normalization technique to scale down the dataset to [0, 1]. The min–max normalization also removes the effect of outlier points. If for some cases, the outlier points carry some important information, then the z-score minimization technique may be used to make the dataset well-structured (Sharma and Verma, 2021).

4.3. Dataset preparation

For the study of fault diagnosis with the proposed GS-EvoN2N, we prepare the training, the validation, & the testing dataset under various operating conditions described below.

Case-1 (T1): From *Air Compressor Dataset*, 7 different cases of binary classes and one case of multi-class diagnosis are investigated as listed in Table 1. For each class, 4 measurement files (having 50k samples/file) are merged to create a sample of size 1000 × 200 per class taking 200 points as segment length.

Case-2 (T2): From *CWRU FE Dataset*, multi-class diagnosis with class name healthy (**H**), inner race (**IR**), outer race (**OR**), and ball element (**B**) are considered under different load (1, 2, & 3 hp) conditions and different fault diameters (FD) (7, 14, & 21 mil). For each FD (for example, 7 mil), dataset from all three load conditions are prepared. Thus, the fault diagnosis on total of 9 cases are presented as listed in Table 2.

Case-3 (T3 & T4): From *PBU Dataset*, two different cases are considered (i) T3: artificially damaged bearing fault and (ii) T4: bearing fault due to long accelerated test. In both cases, multi-class diagnosis with three classes namely **H-OR-IR** is studied under four load settings (LS) as listed in Table 3.

Now for the training, the validation, & the testing, each of the above dataset is split into three portions: 64% train (D^{tr}), 20% test (D^{te}), and 16% validate (D^{val}) datasets.

4.4. Implementation details

For the implementation of the proposed framework of GS-EvoN2N, the initial parameters are selected as: population size (N) = 100, crossover probability (P_c) = 0.5, and the maximum number of generations is set to very high usually at 50. Also, the termination criteria are set as either the validation accuracy reaches 100% or it does not change continuously for 3 generations. The allowable ranges for the variation of depth and width are selected as $n_R \in [1,\ 10]$ and $h_R \in [10,\ 400]$ respectively. The learning rate α = 0.1. The GS-EvoN2N framework is applied to the training dataset and the best model obtained is tested for the test dataset under all cases (T1, T2, T3, & T4) described in Section 4.3. The classification accuracies (CA) are tabulated in Tables 1, 2, & 3.

Table 1
Air compressor dataset (T1): Diagnostic performance in term of classification accuracy.

Class	SVM (Widodo and Yang, 2007)	DNN (Qi et al., 2017)	DTL (Wen et al., 2019)	DAFD (Lu et al., 2017)	N2N (Sharma and Verma, 2021)		EvoDCNN (Sun et al., 2020)	EvoN2N (Sharma and Verma, 2022)	**GS-EvoN2N**
					W. D. A.	D. A.			
H-LIV	99.75	96.25	99.00	99.75	99.50	99.75	100.00	100.00	**100.00**
H-LOV	98.25	95.75	99.25	99.66	99.25	99.25	99.75	99.75	**100.00**
H-PR	98.25	93.25	93.30	98.75	97.75	98.75	98.25	99.75	**99.75**
H-B	98.25	98.50	98.75	98.75	96.75	98.75	98.75	99.75	**100.00**
H-F	99.25	99.25	99.00	98.75	99.25	99.25	99.25	100.00	**100.00**
H-NRV	98.75	99.00	99.00	99.75	99.00	99.25	99.25	100.00	**100.00**
H-RB	98.25	98.25	98.25	99.00	99.75	99.75	99.25	100.00	**100.00**
H-ALL	97.75	99.25	99.00	99.00	99.25	99.25	99.75	99.75	**100.00**
S. D.	0.65	2.16	2.00	0.46	1.02	0.38	0.57	0.13	**0.09**

Table 2
CWRU FE Dataset (T2): Diagnostic performance in term of classification accuracy.

Class	FD	Load	SVM (Widodo and Yang, 2007)	DNN (Qi et al., 2017)	DTL (Wen et al., 2019)	DAFD (Lu et al., 2017)	N2N (Sharma and Verma, 2021)		EvoDCNN (Sun et al., 2020)	EvoN2N (Sharma and Verma, 2022)	**GS-EvoN2N**
							W. D. A.	D. A.			
H-IR-OR-B	7 mil	1 hp	88.12	96.69	96.56	97.94	98.94	98.94	99.60	100.00	**100.00**
		2 hp	98.12	95.94	93.44	96.12	97.12	98.12	99.60	100.00	**100.00**
		3 hp	99.10	98.75	98.75	98.44	99.44	99.44	99.70	100.00	**100.00**
	14 mil	1 hp	99.10	94.75	96.88	97.19	99.19	99.67	100.00	100.00	**100.00**
		2 hp	98.10	95.31	92.19	95.69	97.69	98.69	98.12	99.12	**99.60**
		3 hp	99.25	96.88	94.69	97.62	99.33	98.62	98.44	98.84	**100.00**
	21 mil	1 hp	96.88	86.56	84.69	89.62	95.62	96.62	93.75	98.75	**100.00**
		2 hp	88.44	85.31	82.19	86.69	90.69	90.69	90.10	95.37	**98.85**
		3 hp	92.19	86.56	79.38	88.06	91.06	92.06	92.81	95.81	**98.81**
S. D.			4.62	5.25	7.06	4.66	3.46	3.32	3.69	1.81	**0.51**

Table 3
CA for Target-2 & Target-3 Dataset and for very limited samples of Target-2 & Target-3 Dataset.

Class	Data-L.S.	SVM (Widodo and Yang, 2007)	DNN (Qi et al., 2017)	DTL (Wen et al., 2019)	DAFD (Lu et al., 2017)	N2N (Sharma and Verma, 2021)		EvoDCNN (Sun et al., 2020)	EvoN2N (Sharma and Verma, 2022)	**GS-EvoN2N**
						W. D. A.	D. A.			
H-OR-IR	T3-L1	94.25	96.92	96.92	96.92	98.64	98.94	99.25	99.75	**100.00**
	T3-L2	90.00	93.58	95.00	94.58	95.12	96.12	99.58	99.83	**99.75**
	T3-L3	87.17	91.92	93.33	92.08	94.44	94.44	97.50	97.70	**100.00**
	T3-L4	87.17	93.15	93.75	94.17	97.19	95.28	100.00	100.00	**100.00**
	T4-L1	95.00	97.50	97.50	98.33	98.33	99.17	99.17	100.00	**100.00**
	T4-L2	92.83	95.92	96.50	96.33	96.33	96.33	98.60	99.15	**100.00**
	T4-L3	94.83	94.67	93.33	94.17	95.72	96.33	98.60	99.15	**100.00**
	T4-L4	94.83	95.33	95.00	95.83	95.69	95.69	93.33	98.75	**99.75**
S. D.		3.41	1.92	1.65	1.95	1.50	1.68	2.13	0.79	**0.10**

4.5. Comparison and results

Since our method deals with fault diagnosis with NAS, the main focus has been on finding the solution for faster selection of the best DNN model for fault diagnosis. Therefore, we compare our results with the baseline methods and the state-of-the-art methods used for fault diagnosis. The state-of-the-art methods for intelligent fault diagnosis best reported in various literature are support vector machines (SVM) (Widodo and Yang, 2007), deep neural network (DNN) (Qi et al., 2017), deep transfer learning (DTL) based on sparse autoencoder (Wen et al., 2019), Deep neural network for domain Adaptation in Fault Diagnosis (DAFD) (Lu et al., 2017), Net2Net without domain adaptation (N2N_WDA) (Sharma and Verma, 2021), Net2Net with domain adaptation (N2N_DA) (Sharma and Verma, 2021), evolutionary deep CNN (EvoDCNN) (Sun et al., 2020), and evolutionary Net2Net (EvoN2N) (Sharma and Verma, 2022). The DNN, DTL, and DAFD are trained with hidden sizes of (70−50−20). The initial and hyper parameters for EvoDCNN and EvoN2N are kept same as mentioned above. The same dataset (T1, T2, T3, & T4) are used to train and test all these methods using the procedure suggested in the corresponding references cited. The diagnostic performance in term of CA are tabulated in Tables 1, 2, & 3. The standard deviation (S.D.) of CA calculated over the variation in the operating conditions is also tabulated to compare the result deviation with the change in the operating conditions.

4.6. Ablation study

The main contribution of this research work is the guided-sampling of population for the evolution of the network architecture. Model architecture sampling for population and offspring generation requires mean and variance terms that are produced by a reward-based controller. The reward-based controller ensures that the mean and variance are updated in the direction to obtain faster optimality. Therefore, the ablation study to show the impact of the key parameters of the proposed method on diagnostic performance is presented in the following steps.

4.6.1. Effect of guided-sampling of population

If guided-sampling is absent in the algorithm, population has to randomly initialized in the search space for offspring generation. Then, the evolution process becomes a completely heuristic search, which is the same as EvoN2N (Sharma and Verma, 2022). The performance results for EvoN2N for the same dataset are provided in Tables 1, 2, 3. Also, the evolution of the best model at every 5th generation has been presented in Table 4 with and without guided sampling. It can be observed that without guided sampling (EvoN2N Sharma and Verma, 2022) takes more number of generations to reach the best model architecture. However, in both cases, the final models obtained

Table 4
Evolution of best architecture (**Hidden Layers and Nodes**) at Every 5th Generation (**CWRU FE, 7 mil FD, 1 hp load**).

#Gen	EvoN2N (No Guided-Sampling) (Sharma and Verma, 2022)				GS-EvoN2N			
	Pop. Size = 50		Pop. Size = 100		Pop. Size = 50		Pop. Size = 100	
	Best Arch.	% CA	Best Arch.	% CA	Best Arch.	% CA	Best Arch.	% CA
0	100-20	89.8	90-88-73	90	95-77-45	91.5	100-82-40	90.1
5	98-95-61	94.3	100-95-75-73-64–35	94.4	90-77-38	93.9	97-80-70-63	94.93
10	100-76-66	95.9	100–45	96.1	91-84-44	95.8	100-83-60-45	97.57
15	99-86-81-74	98.1	97-75-75-60	99.1	100-88-80-71	99.2	**100-100-80-73**	**100**
20	100-96-84-63	99.7	**100-100-80-73**	**100**	100-91-76-52	99.6	**100-100-80-73**	**100**

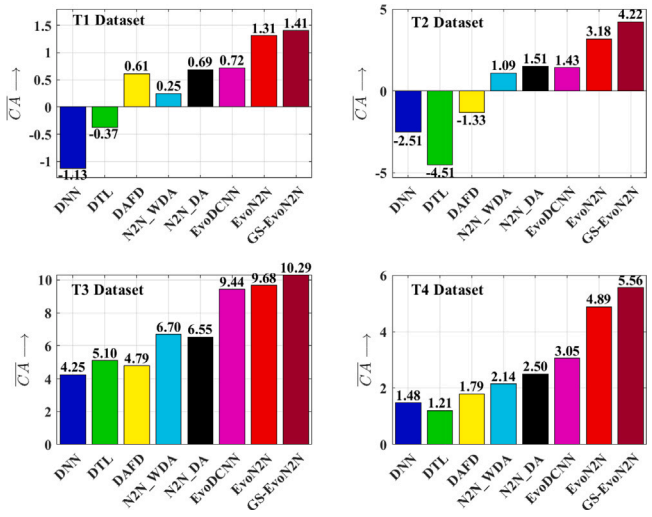


Fig. 6. TI in term of $\overline{CA}$ for (i) T1: Air Compressor dataset, (ii) T2: CWRU dataset, (iii) T3: PBU dataset with single point fault, and (iv) T4: PBU dataset with distributed fault.

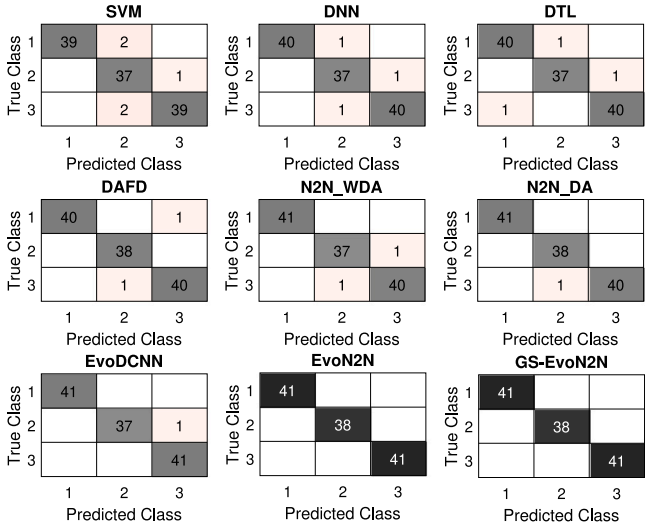


Fig. 7. Confusion matrix for dataset T4-L1 (Table 3): class label {'1', '2', '3'} represents the class name {'H', 'OR', 'IR'}.

are the same. The effect of guided sampling can be observed more clearly from the classification accuracy curve shown in Fig. 8.

4.6.2. Effect of population size

Table 4 shows the architecture produced at various generation by the two methods; the EvoN2N and the GS-EvoN2N for population size 50 and 100. It can be seen that neither of the algorithm converge fully in 20 generations for population size of 50. However, with population

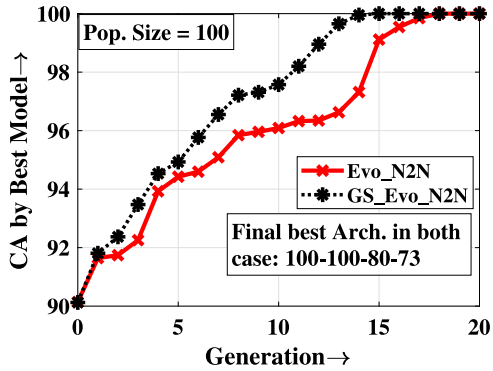


Fig. 8. Rise of CA curve of GS-EvoN2N and EvoN2N for Population Size = 100 (**CWRU FE, 7 mil FD, 1 hp load**).

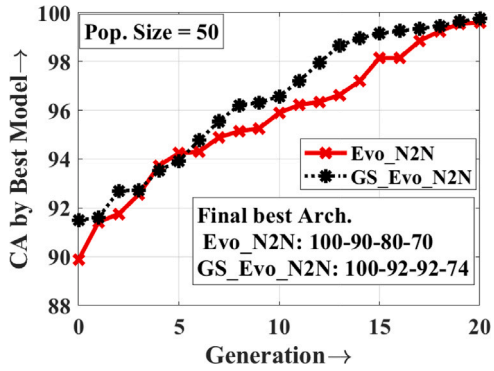


Fig. 9. Rise of CA curve of GS-EvoN2N and EvoN2N For Population Size = 50 (**CWRU FE, 7 mil FD, 1 hp load**).

size = 100, both the algorithm converge fully to a model that provides 100% classification accuracy on the validation data. Therefore, small population size may require more number of generations to converge or may not converge. The rise of the accuracy curve for the evolutionary architecture search with and without guided-sampling is shown in Fig. 9. It can be observed that both the curves may converge to produce 100% accuracy for a higher number of generation.

4.6.3. Effect of number of generation

From Table 4 and accuracy curve shown in Fig. 8, it can be observed that the proposed method takes about 12 epochs to reach to its optimal point where as the evolution without guided-sampling takes about 18 epochs to reach to the optimal architecture. If the population size is reduced by half, the number of generations required are significantly increased.

4.7. Discussion

The diagnostic performances of the proposed method and the selected state-of-the-art methods conclude the following observations.

(i) The CA comparison in Tables 1, 2, & 3 reveal that diagnostic performances are very much affected by the best architecture selection. The DNN model with the best suitable architecture for the given dataset can perform up to almost 100% CA while other methods with pre-selected architecture fail to perform well.

(ii) Considering SVM (Widodo and Yang, 2007) as the baseline diagnostic method, We evaluate the transfer improvement (TI) in terms of average CA for the dataset T1, T2, T3, & T4 separately. If the average CA is denoted as $\overline{CA}$, the TI is defined as $TI = \overline{CA} - \overline{CA}_b$, where $\overline{CA}_b$ is the average CA by SVM. The TI graph shown in Fig. 6 shows the performance improvement of the proposed framework compared to the state-of-the-art methods and the baseline method 'SVM'.

(iii) Fig. 7 demonstrates the classification performance by confusion chart matrices for one of the dataset (T4-L1: Table 3). The confusion matrices with blackened diagonal elements represent the classifications with 100% accuracies. The confusion matrices with gray shades are the methods with missed classifications. All the test samples are correctly classified using the proposed method GS-EvoN2N, therefore, the proposed method is capable of selecting the best possible architecture that has almost 100% diagnostic performance.

(iv) Fig. 8 shows the evolution of the best model for the proposed GS-EvoN2N and EvoN2N (Sharma and Verma, 2022) with generation. The comparison of the rise of the curve reveals that guided sampling helps the algorithm to attain optimality faster. Therefore, the proposed method GS-EvoN2N requires fewer generations to converge to the global optima.

(v) Table 4 shows the evolution of the best architecture in every 5th generation. It can be seen that the proposed GS-EvoN2N reached its optimality before 15th generation much faster compared to EvoN2N. Also, it can be observed that the best architecture may be having larger or smaller size. The final best architecture has a small size compared to what it was obtained at 5th generation. Therefore, only increasing the size of the architecture cannot perform better for fault diagnosis of industrial machines with limited availability of training samples.

(vi) The comparison between EvoDCNN (Sun et al., 2020), EvoN2N (Sharma and Verma, 2022), & the proposed GS-EvoN2N reveals that the fully connected model (DNN) with the best architecture is more suitable for fault diagnosis applications compared to the CNN model in EvoDCNN (Sun et al., 2020) proposed for image classification.

(vii) The time-series data classification by manually designed transformer models (Wen et al., 2023) may be very effective solution for fault diagnosis problems due to their excellent capability of capturing long-term dependencies. But, transformer models require large number of training samples which may be not be feasible for our case due to limited availability of the training samples. Furthermore, the number of trainable parameters in each transformer block with 768 hidden size would be much more than the best model obtained in our case (**100-100-80-73**).

4.8. Complexity analysis

The worst complexities in one iteration of the entire algorithm 1 are contributed by (i) fitness evaluation of the DNN model and (ii) the non-dominated sorting. The complexity of the fitness evaluation of DNN is mainly contributed by parameter optimization by L-BFGS which is $O(N_I * n^2)$, where n & N_I be the total number of parameters and number of iterations required to fine-tune the DNN model. The non-dominated sorting algorithm has a complexity of $O(MN_p^2)$, where M & N_p^2 are the number of objectives and the population size, respectively. Since M is very small compared to N_I, therefore, the time complexity for GS-EvoN2N with population size N_p is given by $O(N_I N_p n^2)$, where n = total number of parameters in one model and N_I = number of iterations taken for model training.

Since, the fitness evaluation adopted in the proposed framework use transfer of knowledge of best model from the previous generation, the fitness evaluation gets faster and cheaper with generation. The parameter optimization by L-BFGS requires lesser number of epochs as the evolution proceeds towards its optimality. Therefore, carbon footprint for the proposed framework is less than the state-of-the-art evolutionary NAS. Furthermore, since the time-series recorded signals are segmented as small-size samples, fully connected network having architecture like (100-100-80-73) performs well, the embedded devices with modern CPUs would be sufficient for real-time deployment.

5. Conclusions

In this article, we have formulated a guided sampling-based evolutionary algorithm for the search of DNN architecture. The proposed framework uses the concept of policy gradient to sample the new population to force the evolution towards the maximization of classification performance. The classification accuracies of the best model in each generation are used as a reward to update the policy parameters. The policy controller generates the mean and variance term to sample the new architecture for better performance. The best model obtained in each generation is also transferred to the next generation to initialize the model evaluation using the concept of net2net transformation. The entire algorithm becomes faster to attain the global maxima. Therefore, this method is very good in terms of faster evolution and faster convergence while ensuring global convergence. The validation using dataset under various cases from Air Compressor data, CWRU data, and PBU data proves that the proposed framework is capable of obtaining the best model to get diagnostic performance almost up to 100% accuracy. This method can also be used for the architecture optimization of the CNN model with image classification or object detection applications.

CRediT authorship contribution statement

Arun K. Sharma: Conceptualization, Methodology, Software implementation, Writing. **Nishchal K. Verma:** Guidance, Reviewing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data are openly available and reference to the data has been added to the reference section.

References

Baker, B., Gupta, O., Naik, N., Raskar, R., 2016. Designing neural network architectures using reinforcement learning. CoRR abs/1611.02167.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., 2007. Greedy layer-wise training of deep networks. In: Schölkopf, B., Platt, J.C., Hoffman, T. (Eds.), Advances in Neural Information Processing Systems 19. MIT Press, pp. 153–160.

Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B., 2011. Algorithms for hyper-parameter optimization. In: Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 24. Curran Associates, Inc., pp. 2546–2554.

Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, 281–305.

Chen, Z., Han, F., Wu, L., Yu, J., Cheng, S., Lin, P., Chen, H., 2018a. Random forest based intelligent fault diagnosis for PV arrays using array voltage and string currents. Energy Convers. Manage. 178, 250–264.

Chen, Y., Meng, G., Zhang, Q., Xiang, S., Huang, C., Mu, L., Wang, X., 2019. Renas: Reinforced evolutionary neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4787–4796.

Chen, X., Wang, S., Qiao, B., Chen, Q., 2018b. Basic research on machinery fault diagnostics: Past, present, and future trends. Front. Mech. Eng. 13, 264–291.
Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197.
Dudziak, Ł., Chau, T., Abdelfattah, M.S., Lee, R., Kim, H., Lane, N.D., 2020. BRP-NAS: Prediction-based NAS using GCNs. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20, Curran Associates Inc., Red Hook, NY, USA.
Fan, L., Wang, H., 2023. Surrogate-assisted evolutionary neural architecture search with network embedding. Complex Intell. Syst. 9 (3), 3313–3331.
Fan, W., Zhou, Q., Li, J., Zhu, Z., 2018. A wavelet-based statistical approach for monitoring and diagnosis of compound faults with application to rolling bearings. IEEE Trans. Autom. Sci. Eng. 15 (4), 1563–1572.
Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., Lempitsky, V., 2016. Domain-adversarial training of neural networks. J. Mach. Learn. Res. 17 (59), 1–35.
Guo, L., Lei, Y., Xing, S., Yan, T., Li, N., 2019. Deep convolutional transfer learning network: A new method for intelligent fault diagnosis of machines with unlabeled data. IEEE Trans. Ind. Electron. 66 (9), 7316–7325.
He, Q., Liu, Y., Fanrang, K., 2010. Machine fault signature analysis by midpoint-based empirical mode decomposition. Meas. Sci. Technol. 22, 015702.
He, X., Zhao, K., Chu, X., 2021. Automl: A survey of the state-of-the-art. Knowl.-Based Syst. 212, 106622.
Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science 313 (5786), 504–507.
Hundt, A., Jain, V., Hager, G.D., 2019. Sharpdarts: Faster and more accurate differentiable architecture search. CoRR abs/1903.09900. [Online]. Available: http://arxiv.org/abs/1903.09900.
Hutter, F., Hoos, H.H., Leyton-Brown, K., 2011. Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (Ed.), Learning and Intelligent Optimization. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 507–523.
Jaafra, Y., Luc Laurent, J., Deruyver, A., Saber Naceur, M., 2019. Reinforcement learning for neural architecture search: A review. Image Vis. Comput. 89, 57–66.
Juan Jose, S.D., et al., 2016. Multifault diagnosis method applied to an electric machine based on high-dimensional feature reduction. IEEE Trans. Ind. Appl. 53 (3), 3086–3097.
Kandasamy, K., Neiswanger, W., Schneider, J., Póczos, B., Xing, E.P., 2018. Neural architecture search with Bayesian optimisation and optimal transport. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. NIPS '18, pp. 2020–2029.
Lei, Y., He, Z., Zi, Y., 2008. A new approach to intelligent fault diagnosis of rotating machinery. Expert Syst. Appl. 35 (4), 1593–1600.
Lessmeier, C., Kimotho, J., Zimmer, D., Sextro, W., 2016. Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification. European Conf., PHM Society, Bilbao (Spain) 3 (1).
Li, J., Cao, X., Chen, R., Zhang, X., Huang, X., Qu, Y., 2023a. Graph neural network architecture search for rotating machinery fault diagnosis based on reinforcement learning. Mech. Syst. Signal Process. 202, 110701.
Li, X., Xu, Y., Li, N., Yang, B., Lei, Y., 2023b. Remaining useful life prediction with partial sensor malfunctions using deep adversarial networks. IEEE/CAA J. Autom. Sin. 10 (1), 121–134.
Li, X., Yu, S., Lei, Y., Li, N., Yang, B., 2023c. Intelligent machinery fault diagnosis with event-based camera. IEEE Trans. Ind. Inform. 1–10.
Li, X., Zhang, W., Ding, Q., 2019. Cross-domain fault diagnosis of rolling element bearings using deep generative neural networks. IEEE Trans. Ind. Electron. 66 (7), 5525–5534.
Liu, Y., Li, X., Hu, Y., 2023. Differentiable neural architecture search for domain adaptation in fault diagnosis. Mech. Syst. Signal Process. 202, 110639.
Liu, H., Simonyan, K., Yang, Y., 2019. Darts: Differentiable architecture search. In: International Conference on Learning Representations.
Liu, S., Zhang, H., Jin, Y., 2022. A survey on computationally efficient neural architecture search. J. Autom. Intell. 1 (1), 100002.
Loghmanian, S.M.R., Jamaluddin, H., Ahmad, R., Yusof, R., Khalid, M., 2012. Structure optimization of neural network for dynamic system modeling using multi-objective genetic algorithm. Neural Comput. Appl. 21 (6), 1281–1295.
Long, M., Wang, J., Ding, G., Pan, S.J., Yu, P.S., 2014. Adaptation regularization: A general framework for transfer learning. IEEE Trans. Knowl. Data Eng. 26 (5), 1076–1089.
Loni, M., Daneshtalab, M., Sjodin, M., 2018. ADONN: Adaptive design of optimized deep neural networks for embedded systems. In: 2018 21st Euromicro Conference on Digital System Design (DSD). IEEE Computer Society, Los Alamitos, CA, USA, pp. 397–404.
Loni, M., Sinaei, S., Zoljodi, A., Daneshtalab, M., Sjödin, M., 2020. DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems. Microprocess. Microsyst. 73, 102989.
Loni, M., Zoljodi, A., Majd, A., Ahn, B.H., Daneshtalab, M., Sjödin, M., Esmaeilzadeh, H., 2022. FastStereoNet: A fast neural architecture search for improving the inference of disparity estimation on resource-limited platforms. IEEE Trans. Syst. Man Cybern.: Syst. 52 (8), 5222–5234.
Loni, M., Zoljodi, A., Sinaei, S., Daneshtalab, M., Sjödin, M., 2019. NeuroPower: Designing energy efficient convolutional neural network architecture for embedded systems. In: Artificial Neural Networks and Machine Learning – ICANN 2019: Theoretical Neural Computation. Springer International Publishing, Munich, Germany.
Lu, Z., Cheng, R., Huang, S., Zhang, H., Qiu, C., Yang, F., 2022. Surrogate-assisted multiobjective neural architecture search for real-time semantic segmentation. IEEE Trans. Artif. Intell. 1–14.
Lu, W., Liang, B., Cheng, Y., Meng, D., Yang, J., Zhang, T., 2017. Deep model based domain adaptation for fault diagnosis. IEEE Trans. Ind. Electron. 64 (3), 2296–2305.
Lu, Z., Whalen, I., Dhebar, Y., Deb, K., Goodman, E.D., Banzhaf, W., Boddeti, V.N., 2021. Multiobjective evolutionary design of deep convolutional neural networks for image classification. IEEE Trans. Evol. Comput. 25 (2), 277–291.
Luo, R., Tian, F., Qin, T., Chen, E., Liu, T.-Y., 2018. Neural architecture optimization. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Vol. 31. Curran Associates, Inc..
Nandi, S., Toliyat, H.A., Li, X., 2005. Condition monitoring and fault diagnosis of electrical motors—A review. IEEE Trans. Energy Convers. 20 (4), 719–729.
Nocedal, J., Wright, S.J., 2006. Large-scale unconstrained optimization. In: Numerical Optimization. Springer New York, New York, NY, pp. 164–192.
Pan, S.J., Tsang, I.W., Kwok, J.T., Yang, Q., 2011. Domain adaptation via transfer component analysis. IEEE Trans. Neural Netw. 22 (2), 199–210.
Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J., 2018. Efficient neural architecture search via parameters sharing. In: International Conference on Machine Learning. PMLR, pp. 4095–4104.
Qi, Y., Shen, C., Wang, D., Shi, J., Jiang, X., Zhu, Z., 2017. Stacked sparse autoencoder-based deep network for fault diagnosis of rotating machinery. IEEE Access 5, 15066–15079.
Qiu, Z., Bi, W., Xu, D., Guo, H., Ge, H., Liang, Y., Lee, H.P., Wu, C., 2023. Efficient self-learning evolutionary neural architecture search. Appl. Soft Comput. 146, 110671.
Real, E., Aggarwal, A., Huang, Y., Le, Q.V., 2019. Regularized evolution for image classifier architecture search. Proc. AAAI Conf. Artif. Intell. 33 (01), 4780–4789, [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4405.
Sharma, A.K., Singh, V., Verma, N.K., Liu, J., 2018. Condition based monitoring of machine using mamdani fuzzy network. In: 2018 Prognostics and System Health Management Conference (PHM-Chongqing). pp. 1159–1163.
Sharma, A.K., Verma, N.K., 2021. Quick learning mechanism with cross-domain adaptation for intelligent fault diagnosis. IEEE Trans. Artif. Intell. 1, (Early Access).
Sharma, A.K., Verma, N.K., 2022. Knowledge transfer based evolutionary deep neural network for intelligent fault diagnosis. arXiv:2109.13479v2 [eess.SP].
Siddique, A., Yadava, G.S., Singh, B., 2005. A review of stator fault monitoring techniques of induction motors. IEEE Trans. Energy Convers. 20 (1), 106–114.
Smith, W.A., Randall, R.B., 2015. ROLLING element bearing diagnostics using the Case Western Reserve University data: A benchmark study. Mech. Syst. Signal Process. 64, 100–131.
Su, H., Chong, K.T., 2007. Induction machine condition monitoring using neural network modeling. IEEE Trans. Ind. Electron. 54 (1), 241–249.
Sun, J., Liu, X., Bäck, T., Xu, Z., 2021a. Learning adaptive differential evolution algorithm from optimization experiences by policy gradient. IEEE Trans. Evol. Comput. 25 (4), 666–680.
Sun, Y., Sun, X., Fang, Y., Yen, G.G., Liu, Y., 2021b. A novel training protocol for performance predictors of evolutionary neural architecture search algorithms. IEEE Trans. Evol. Comput. 25 (3), 524–536.
Sun, Y., Wang, H., Xue, B., Jin, Y., Yen, G.G., Zhang, M., 2019. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. IEEE Trans. Evol. Comput. 24 (2), 350–364.
Sun, Y., Xue, B., Zhang, M., Yen, G.G., 2019. A particle swarm optimization-based flexible convolutional autoencoder for image classification. IEEE Trans. Neural Netw. Learn. Syst. 30 (8), 2295–2309.
Sun, Y., Xue, B., Zhang, M., Yen, G.G., 2020. Evolving deep convolutional neural networks for image classification. IEEE Trans. Evol. Comput. 24 (2), 394–407.
Sutton, R., Barto, A., 2018. Reinforcement Learning: An Introduction, second ed. In: Adaptive Computation and Machine Learning series, MIT Press.
Verma, N.K., Sevakula, R.K., Dixit, S., Salour, A., 2016. Intelligent condition based monitoring using acoustic signals for air compressors. IEEE Trans. Reliab. 65 (1), 291–309.
Wang, R., Jiang, H., Li, X., Liu, S., 2020. A reinforcement neural architecture search method for rolling bearing fault diagnosis. Measurement 154, 107417.
Wang, C., Xu, C., Yao, X., Tao, D., 2019. Evolutionary generative adversarial networks. IEEE Trans. Evol. Comput. 23 (6), 921–934.
Wei, D., Han, T., Chu, F., Zuo, M.J., 2021. Weighted domain adaptation networks for machinery fault diagnosis. Mech. Syst. Signal Process. 158, 107744.
Wen, L., Gao, L., Li, X., 2019. A new deep transfer learning based on sparse auto-encoder for fault diagnosis. IEEE Trans. Syst. Man Cybern.: Syst. 49 (1), 136–144.
Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., Sun, L., 2023. Transformers in time series: A survey.

Widodo, A., Yang, B.-S., 2007. Support vector machine in machine condition monitoring and fault diagnosis. Mech. Syst. Signal Process. 21 (6), 2560–2574.

Williams, R.J., May 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. 8 (3–4), 229–256.

Wöhrle, H., Schneider, F., Schlenke, F., Lebold, D., De Lucas Alvarez, M., Kirchner, F., Karagounis, M., 2023. Multi-objective surrogate-model-based neural architecture and physical design co-optimization of energy efficient neural network hardware accelerators. IEEE Trans. Circuits Syst. I. Regul. Pap. 70 (1), 40–53.

Yan, X., Jia, M., 2018. A novel optimized SVM classification algorithm with multi-domain feature and its application to fault diagnosis of rolling bearing. Neurocomputing 313, 47–64.

Yang, Z., Wang, Y., Chen, X., Shi, B., Xu, C., Xu, C., Tian, Q., Xu, C., 2020. Cars: Continuous evolution for efficient neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1829–1838.

Yin, S., Ding, S.X., Xie, X., Luo, H., 2014. A review on basic data-driven approaches for industrial process monitoring. IEEE Trans. Ind. Electron. 61 (11), 6418–6428.

Yu, X., Dong, F., Ding, E., Wu, S., Fan, C., 2017. Rolling bearing fault diagnosis using modified LFDA and EMD with sensitive feature selection. IEEE Access 6, 3715–3730.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., Gao, R.X., 2019. Deep learning and its applications to machine health monitoring. Mech. Syst. Signal Process. 115, 213–237.

Zoph, B., Le, Q.V., 2016. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.

Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V., 2018. Learning transferable architectures for scalable image recognition. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.. pp. 8697–8710.