Journal of Big Data

RESEARCH Open Access

Automatic DNN architecture design using CPSOTJUTT for power system inspection

Xian-Long Lv[1], Hsiao-Dong Chiang[2] and Na Dong[1*]

*Correspondence:
dongna@tju.edu.cn

[1] School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China
[2] School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853, USA

Abstract

To quickly and accurately automatically design more high-precision deep neural network models (DNNs), this paper proposes an automatic DNN architecture design ensemble model based on consensus particle swarm optimization-assisted trajectory unified and TRUST-TECH (CPSOTJUTT), called CPSOTJUTT-EM. The proposed model is a three-layer model, and its core is a three-stage method for addressing the sensitivity of the local solver to the initial point and enabling fast and robust training DNN, effectively avoiding missing high-quality DNN models in the process of automatic DNN architecture design. CPSOTJUTT has the following advantages: (1) high-quality local optimal solutions (LOSs) and (2) robust convergence against random initialization. CPSOTJUTT-EM consists of the bottom layer: stable and fast design high-quality DNN architectures, the middle layer: exploration for a diverse set of optimal DNN classification engines, and the top layer: ensemble model for higher performance. This paper tests the performance of CPSOTJUTT-EM on public datasets and three self-made power system inspection datasets. Experimental results show that the CPSOTJUTT-EM has excellent performance in automatic DNN architecture design, DNN model optimization. And the CPSOTJUTT-EM can automatically design high-quality DNN ensemble models, laying a solid foundation for the application of DNN in other fields.

Keywords: Deep neural network, CPSOTJUTT, Ensemble model, Consensus-based PSO, Trajectory unified, TRUST-TECH methodology, Power system inspection

Introduction

Deep neural networks (DNNs) are widely used in computer vision, natural language processing, speech recognition, and other fields [1, 2]. The most state-of-the-art DNN models like ResNet [3] and EfficientNet [4] currently mainly rely on manual design based on a common standard dataset, such as ImageNet. However, DNN models usually do not show high performance in many specific field tasks. In order to design a DNN with good performance, it is necessary to have extensive professional knowledge in both the DNN and the problem field being studied, which may not necessarily be applicable to every interested user [5]. To address the above issues, Automatic DNN architecture design technology is an efficient solution that can meet the needs of tasks in different fields [6]. This method is expected to significantly reduce labor costs and improve model performance, promoting the application of DNN in other fields.

In critical and sensitive domains, such as medical diagnosis and autonomous driving, there is an increasing demand for high-precision model. A single model often exhibits different recognition abilities in different categories, which makes it difficult to meet the required accuracy [7]. With the advancement of hardware performance, the application of large-scale ensemble models is becoming more widespread. The ensemble model improves the accuracy and generalization ability of the model by utilizing multiple models to comprehensively consider the prediction results through joint learning and decision-making [8]. However, designing and optimizing the ensemble model requires in-depth learning of professional knowledge and skills, including model selection and ensemble method design, which affects its application in other fields [9].

This paper combines automatic DNN architecture design technology and ensemble model technology to achieve improved DNN performance. Automatic DNN architecture design technology can automatically search and generate DNN models that are more suitable for specific tasks and domain requirements. Once multiple proposal optimized DNN architectures are obtained, ensemble model technology can synthesize the prediction results of these models to achieve higher accuracy. However, the high redundancy and nonconvexity of the parameters lead to many local optimal solutions (LOSs) for the DNN. Also, both low-quality and high-quality LOSs all have the same local properties [10, 11]. DNN training is usually realized by the first-order local solver [12]. One disadvantage of the local solver is that the gradients in different directions are uniformly scaled and may converge to a bad LOS [13, 14], resulting in poor generalization ability or inability to converge [15]. For the poor robustness of the training DNN method, the performance of the automatically designed DNN architecture cannot be adequately evaluated.

To solve the above problems, this paper proposes a novel three-layer ensemble model, termed consensus particle swarm optimization-assisted trajectory unified and TRUST-TECH ensemble model (CPSOTJUTT-EM). This model is based on automatic DNN architecture design technology, with CPSOTJUTT algorithm as the core. The bottom layer of CPSOTJUTT-EM achieves stable and fast generation of high-quality DNN architecture. Through this design, the experts of non-deep learning fields can also design the most suitable DNN framework for different domain requirements, effectively promoting the application of deep learning in other fields. The middle layer utilizes a three-stage method for high-precision and robust DNN training to obtain candidate optimized DNN models. The top layer achieves higher performance ensemble model by ensemble high-quality DNN models. The ensemble model can fully leverage the performance advantages of high-quality sub-DNN models, improving the overall accuracy and generalization ability of the model.

The main contributions of this paper are:

1) The CPSOTJUTT-EM can robustly and automatically design a high-quality DNN architecture according to the application field without extended expertise in DNNs.
2) The CPSOTJUTT-EM constructs an ensemble model consisting of a diverse set of high-quality classification engines so that the ensemble model takes full advantage of each sub-DNN to maximize recognition accuracy. The generalization ability of the ensemble model is significantly improved.

3) The CPSOTJUTT methodology with a strong theoretical basis can robustly converge to high-quality LOSs from random initial points.
4) The CPSOTJUTT methodology, which consists of the consensus-based PSO, TJU methodology, and the TRUST-TECH methodology, fully utilizes the global view of the consensus-based PSO, the robust convergence ability of TJU methodology, and the search ability of the TRUST-TECH methodology for higher quality LOSs.

Original contributions and novelties

The architecture of the proposed CPSOTJUTT-EM is given in Fig. 1.

Bottom layer: Automatically designs high-quality DNN architectures. The CPSOTJUTT methodology trains these DNN architectures and selects high-quality DNN classification engines. In this layer, the consensus-based PSO is used to solve the sensitivity of training DNN to the initial value, which can converge quickly to the optimal stability region.

Middle layer: Explore a diverse set of high-quality DNN classification engines via the CPSOTJUTT methodology. The CPSOTJUTT methodology can robustly converge to the LOS and search for better ones nearby while maintaining its global search ability.

Top layer: Take the high-quality classification engines in the middle layer as a hidden node of the ensemble model, and apply the CPSOTJUTT methodology to strengthen the training to find the optimal combination of classification engines to further improve the identification accuracy and generalization ability.

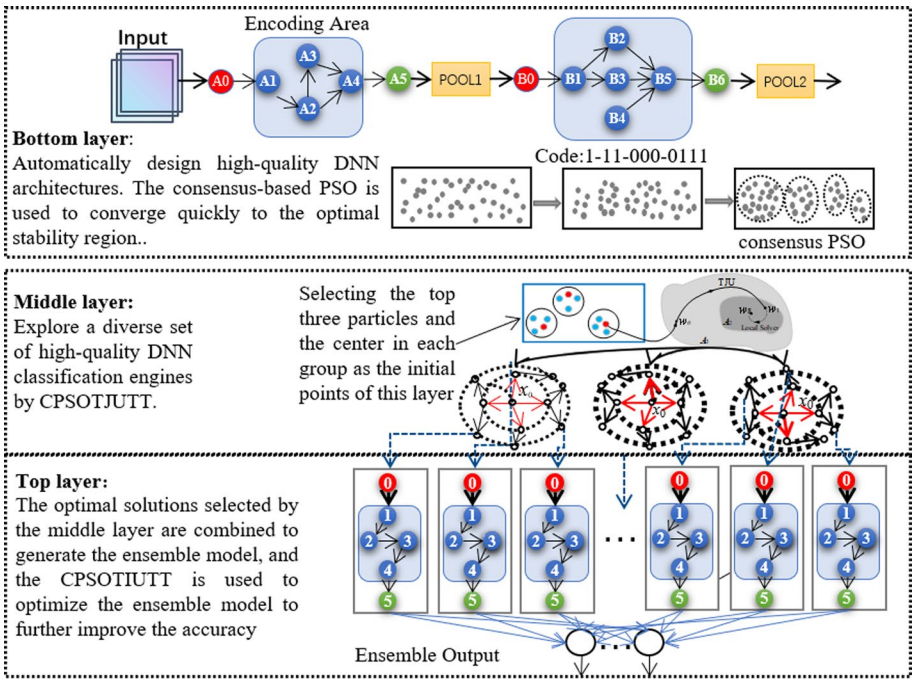

Fig. 1 The architecture of the CPSOTJUTT-EM for power line inspection

Related work

Evolutionary neural network

Several methodologies have been proposed to automatically design DNN architecture. NeuroEvolution of Augmenting Topologies (NEAT) [16] represents early progress in the development of small-scale network architecture, which inspires the research of neuroevolution based on DNN. The NEAT model and co-evolution of modules are combined in CoDeepNEAT [17].

The evolutionary algorithm uses an intuitive mutation operator to add layer structure, which makes the framework complex for a small network [18]. It has made remarkable achievements in the CIFAR dataset. AmoebaNet-A improves the tournament selection evolutionary algorithm by adding an age property that favors the younger genotypes and surpasses hand designs for the first time [19]. The Genetic CNN algorithm [20] is a neuroevolutionary algorithm that optimizes connections between convolutional layers using mutation and cross-evolution. The algorithm can meet the design requirements of the DNN model in some fields to some extent. CNN-GA [21] effectively addresses the image classification tasks by designing a new encoding strategy for the GA to encode arbitrary depths of CNNs. Auto-evolutionary CNN (AE-CNN) [5] provides effective local search and global search ability through a crossover operator and a mutation operator and can design high-quality DNN architectures in the case of limited computing resources. CorrNet is a novel correlation-based pruning (CFP) approach, which creates a feature selection scheme to obtain the pruning approaches. This approach achieves accuracy gain while saving a significant amount of computational costs [22].

Deep neural network training methods

The existing DNN training methods mainly adopt the first-order gradient method and its variants. The optimization algorithm based on a first-order gradient has linear efficiency in time and memory complexity and has achieved great success. Momentum Stochastic Gradient Descent (SGD) [12] pursues fast and stable convergence and is widely used for its simplicity and intuitiveness. However, the gradients in different directions are scaled uniformly, causing poor convergence when training sparse data. Therefore, the acceleration of SGD has attracted extensive research. Recently, some adaptive first-order optimization methods have been proposed to achieve rapid convergence. Adagrad [23] accelerates DNN training by dynamically adjusting the learning rate based on the gradient. RMSprop [24] is an adaptive first-order optimization method that discards remote gradients by using an exponentially declining average of squared gradients. This method has a much lower computational cost than SGD. Adam [25] combines the advantages of Adagrad and RMSprop, which scale the gradient by the square root of the accumulative square gradient to achieve fast convergence. Adam has become the default optimization algorithm for many DNNs due to its rapid convergence [26]. However, due to the sensitivity to initialization and hyperparameters, these optimization methods may converge to sub-optimal local optimal solutions, resulting in worse generalization ability [10].

Deep neural network ensemble

The degree of DNN training has a significant impact on the accuracy and generalization ability of image classification. Insufficient DNN training will result in the low accuracy of hard examples, and overtraining will result in the poor generalization ability of easy examples, especially for the class imbalanced dataset [27, 28]. Cascade structure improves the recognition accuracy of a DNN, but high model capacity will lead to overfitting. The ensemble model of a DNN is the optimal combination of diverse high-quality sub-DNNs, which can fully exploit of the recognition ability of any sub-DNN [29]. Plantdiseasenet uses the majority voting ensemble model to detect plant pests in the early stage of disease, and the results show that the proposed model has reached or exceeded the latest result [30]. The ensemble model of a DNN can obtain better accuracy and generalization ability than each sub-DNN [27].

Power system inspection

As an indispensable infrastructure in modern society, the stable operation of the power system is crucial to the development of social economy and the normal conduct of people's lives. In order to ensure the safe and reliable operation of the power system, regular power system inspections are particularly important.

Power line insulator inspection is a regular assessment of the status of insulators on power transmission lines to ensure the stable and safe operation of the power system. Literature [31] proposes a power insulator inspection algorithm based on deep learning to eliminate the impact of complex power environments on detection accuracy. Power system substation inspection can promptly detect potential equipment failures and take maintenance measures to ensure the safe operation of the power grid. Literature [32] proposes a detection algorithm based on improved YOLO v5. A backbone with a unique attention mechanism was designed to extract more accurate feature maps. Solved the pain point of lack of detection accuracy in unmanned substations. Power line obstacle inspection can identify potential risks and take timely measures to ensure the normal operation of transmission lines. Literature [33] proposed an object detection algorithm based on R-CNN to ensure the safety of power lines.

The automatic DNN architecture design using CPSOTJUTT for power system inspection method proposed in this article improves the accuracy and generalization ability of power system inspection. Solved the problems faced by power system inspection, such as multiple inspection scenarios, low accuracy of general single models, and high difficulty in designing specialized models. In the future, we will conduct research on more advanced deep design network models such as deep learning with prior knowledge [34, 35] to further improve the performance of the proposed methods.

The CPSOTJUTT methodology

The CPSOTJUTT methodology, which consists of the consensus-based PSO, TJU methodology, and Trust-Tech methodology, can converge robustly to high-quality LOSs from random initial points. The CPSOTJUTT methodology is the core of CPSOTJUTT-EM and fully utilizes the global view of the consensus-based PSO, the robust convergence ability of the TJU methodology, and the search ability of the TRUST-TECH methodology

for higher quality LOSs. The pseudocode of the CPSOTJUTT methodology is shown in Algorithm 1.

The architecture of the CPSOTJUTT methodology is as follows:

Stage I: Exploration and Consensus: The positions of the DNN are updated by PSO until PSO is terminated when all the particles have reached a consensus. The three optimal particles in the consensus region and the weight center point are also selected as the initial points of the next stage.

Stage II: Robust Convergence: We use TJU methodology and a local solver to robustly converge to a high-quality LOS from the representative particles selected in the previous stage.

Stage III: Search Optimal: TRUST-TECH methodology is applied to effectively jump out of the stability region of the SEP found in stage II, enter the stability region of neighboring SEPs, and obtain multiple high-quality LOSs in a tier-by-tier search manner.

Algorithm1 Pseudocode of the CPSOTJUTT methodology.	
1:	Initialize the particle swarm, best position *pbest*, and the global best position of the swarm *gbest.*
2:	**repeat**
3:	Update the position $\boldsymbol{P_K}$ and velocity $\boldsymbol{V_K}$ of the swarms.
4:	Update the *pbest* and *gbest*.
5:	**If** not consensus, **then**
6:	Execute mini-batch k-means clustering the particles.
7	**End If**
8:	**until** the consensus-based PSO is satisfied.
9:	The TJU methodology and local solver are used as the local solver to locate the LOSs.
10:	Search the better LOSs by the TRUST-TECH methodology.
11:	Output the high-quality sub-DNN.

CPSOTJUTT stage I: exploration and consensus

The DNN is a regularized version of a multi-layer perceptron with a multi-layer network structure, and its performance is usually evaluated by. The goal of optimal DNN training is to reduce cross-entropy (CE) loss function as much as possible, or even approach 0 infinitely:

$$h(x) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} t_{ij}\log(p_{ij}) \tag{1}$$

where x is the input data, C is the number of classification objects, t_{ij} is the one-hot value of the class, p_{ij} is the probability that sample i belongs to class j, and N is the size of the Mini-Batch.

In this stage, the global search ability of PSOs is used to assist the robust convergence of the local solver. To this end, we introduce a consensus-based PSO to locate optimal converge regions in the search space that contain high-quality LOSs.

All particles exchange information with the personal best position and the global best position at each step of the PSO. The update of the particle is the combination of the original position and velocity, which can be described as follows:

$$V_K = \omega V_{K-1} + C_1 R_1 (pbest - P_{K-1}) + C_2 R_2 (gbest - P_{K-1}) \quad (2)$$

where ω is the weight of the DNN. C_1, C_2 are learning factors, respectively. R_1, R_2 are random numbers distributed between 0 and 1. *pbest* is the personal best position, while *gbest* is the global best position.

The updated personal position is calculated by (3).

$$P_K = P_{K-1} + V_K \quad (3)$$

However, the PSO lacks a global view and fast convergence ability in the later stage of DNN training [36]. To solve the problem of the computational burden of a PSO, we adopt a consensus-based PSO.

The CPSO can locate optimal convergence regions in the search space that contains high-quality LOSs by exchanging information with the personal best position and the global best position [36, 37]. As shown in Fig. 2, all particles will reach a consensus state by converging into one or more regions.

We use mini-batch K-Means [38] to cluster all the particles into several groups at each fixed interval. The mini-batch k-means method can reduce the calculation order of magnitude and has a better clustering performance in high-dimensional optimization problems. The following is the stopping criterion of the consensus-based PSO:

- In the subsequent 5 generations of CPSO, the members of particle groups did not change.

Numerical studies indicate that all particles have good global search ability in the early stage. With the exchange of information among all particles, the global search ability decreases gradually, while the local search ability increases. The PSO algorithm has global optimal particles and more diversity in the consensus state, with a lower computational cost. Thus, we select representative particles in each particle group as the initial point for the next stage of CPSOTJUTT.

CPSOTJUTT stage II: robust convergence

When stage I is completed, the methodology enters stage II, which is the robust convergence stage, as shown in Fig. 3. At this stage, we use the representative particles selected in the previous stage as the initial points w_0 and use the TJU methodology for robust convergence. The TJU methodology has fast and robust convergence during

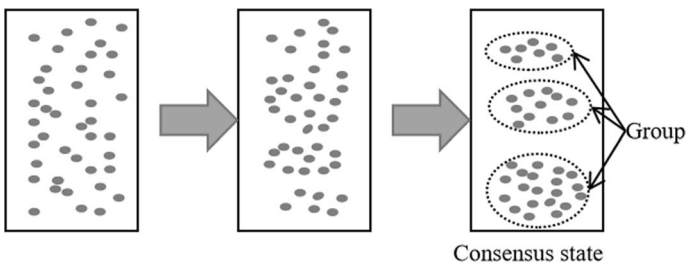

Fig. 2 Process for CPSO to reach a consensus state

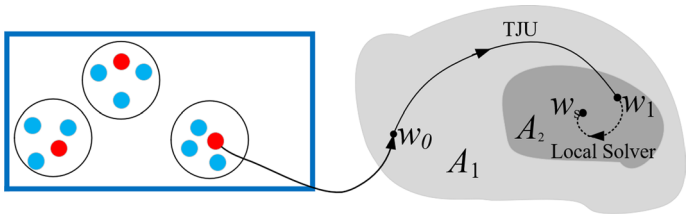

Fig. 3 TJU is used to robustly and accurately compute a LOS. A1 and A2 are stability regions of TJU and the local solver, respectively

the early phase but slows down in the late phase. Therefore, the local solver (such as SGD, Adam) is used to enhance convergence after the TJU methodology.

TJU constructs a dynamical system based on the DNN such that LOSs of the DNN is mapped into SEPs of the dynamical system. Then, by starting from the representative initial point selected in the previous stage, the ensuing trajectory will enter the stability region of the SEP. The following is the nonlinear system of (1):

$$F(w,x) = \begin{bmatrix} h(w,x_1) \\ h(w,x_1) \\ \cdots \\ h(w,x_M) \end{bmatrix}, w \in R^N \tag{4}$$

where h is the loss of the DNN, w is the weight, x is the input data, and N is the size of the mini-batch.

The key of TJU methodology is to construct an effective dynamical system corresponding to the nonlinear system (4) and solve for solutions of (1) via the dynamic trajectories of the constructed nonlinear dynamical system, which can be described as follows:

$$\dot{w} = -\alpha \cdot \nabla F(w)^T \cdot F(w) \tag{5}$$

where $\nabla F(w)$ is the gradient of $F(w)$. When $\sigma = 1$, this is the Focal Loss used by [39]. The system fully considers the gradient information and loss information of the deep neural network model, and mainly focuses training on a sparse set of hard examples.

We apply a technique called the pseudo-transient continuation method (PTC) to realize fast calculation of the steady-state solution. This method can be explained as follows:

$$w = w - (I/\delta - D)^{-1}\dot{w} \tag{6}$$

where w is the weight value, I is the unit matrix, d is the time step, and D is the Jacobian of dynamical system (5).

The training speed can be accelerated with the correction of the dynamical system search direction.

$$\delta_i = \delta_{i-1} \cdot \frac{\|(\dot{w}_{n-1})\|^2}{\|(\dot{w}_n)\|^2} \tag{7}$$

The PTC methodology can reliably compute a small-scale deep neural network model. To improve the scalability of the TJU methodology and train a large-scale

DNN model, the Block-Diagonal-Pseudo-Transient-Continuation (BD-PTC) method is proposed to find the search direction D (D is a $1 \times n$ matrix) [40]:

$$D_{t+1} = D_t - \eta_{t+1} \frac{s_t^T y_t - s_t^T D_t s_t}{tr(s_t^4)} Diag(s_t^2) \tag{8}$$

where $d_t = (I/\delta - D_t)^{-1}\dot{w}$ is the corrected update direction. The $s_t = D_{t-1}^{-1}\dot{w}_t$ and $y_t = \dot{w}_{t+1} - \dot{w}_t$, $\theta_t \in R^n$ denote the parameter to be optimized, $\dot{w}_t \in R^n$ is the dynamical system at θ_t, η_t denotes the step size, and tr denotes the trace operator. $Diag(s_t^2)$ is the diagonal matrix with diagonal elements from the vector s_t.

Take the last (*Nth*) iteration of the BD-PTC methodology as the initial point and apply a local solver to locate a LOS for problem (1).

The following describes the pseudocode of CPSOTJUTT Stage II:

Algorithm 2: The pseudocode of CPSOTJUTT Stage II

Input: m (max of stochastic steps iteration); initial point w_0, set $w = w_0$; Data $x = \{x_1, \ldots, x_B\}$, partitioned into B batches of equal size; nonlinear system H_i and gradient ∇H_i for each batch $i \in [1, B]$; Constants $\zeta = 10^{-4}$ and ε=0.01.

for $e = 1$ **to** m **do**
  for $i = 1$ **to** B **do**
    Calculate $\alpha = -\log(1 - P_i)^{\gamma}$
    Calculate QGS $w_i = -\alpha \nabla F_i(w)^T F_i(w)$
    if first step **then**
      Calculate $\delta_i = 10^{-2}/\left\|\dot{w}_1\right\|^2$
    else then
      $\delta_i = \delta_{i-1} \frac{||(\dot{w}_{n-1})||2}{||(\dot{w}_n)||2}$
    end if
    $y = \dot{w}_i - \dot{w}_{i-1}$
    $\lambda = \frac{s_i^T y_i - s_i^T D_i s_i}{(s_i^T D_i s_i)^2}$
      $D_i = D_{i-1} + \lambda \cdot Diag(s_{i-1}^2)$
      $D_i = clamp(|D_i|, min = \varepsilon)$
    $s_i = (I/\delta_i + D_i)^{-1}\dot{w}_i$
    Update $w_i = w_{i-1} - \eta_i s_i$
  end for
  if $\zeta > \|y_e\|^2$ **then**
    break
  end if
end for
$w = LocalSolver(w,x)$
Output w

CPSOTJUTT stage III: search optimal

The TRUST-TECH methodology can effectively jump out of the stability region of the SEP found in stage II, enter the stability region of neighboring SEPs, and obtain multiple SEPs in a tier-by-tier search manner. This stage has a strong theoretical basis [41].

An intuitive description of the TRUST-TECH methodology is shown in Fig. 4, where $w_{i,j}$ represents different SEPs, i represents the number of tiers of SEPs, and j represents the number of SEPs in this tier. The key steps of the TRUST-TECH methodology are detailed as follows:

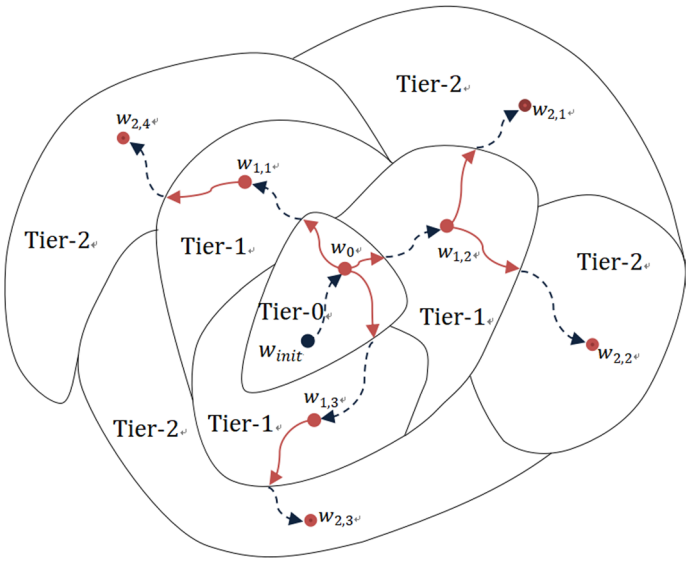

Fig. 4 Schematic diagram of search path of the TRUST-TECH methodology

Step 1 Starting from w_0, step outward in the direction of jumping out of this stability region until the exit point on the stability boundary is reached.

Step 2 Enter the adjacent stability region from the exit point and locate Tier-1 of the stability region.

Step 3 Locate multiple SEPs on Tier-1 by adjusting the direction of jumping out of w_0.

Step 4 Repeat steps 1–3.

Note that the exit point on the stability boundary refers to the point where the loss value changes from ascending gradually to descending steadily, indicating that the trajectory has entered the stability region of another SEP. There is a non-empty intersection point set between the stability boundaries of each tier of SEPs, i.e., the exit point set.

Next, we will describe in detail how the trajectory moves during TRUST-TECH methodology. First, we use a local solver to get a first (Tier-0) SEP w_0. Then we define a trainable search direction g_i, and the parameter vector w_i can be updated by the following equation:

$$\mathrm{w_i} = \mathrm{w_0} + \rho_1(\mathrm{i})\mathrm{g_i} \tag{9}$$

where $\rho_1(i) \in (0, \rho_{max})$ is a learning rate away from w_0, increasing from 0 to ρ_{max}.

When g_i is fixed, the search direction of the parameter vector w_i is fixed. However, in the face of high-dimensional large-scale models, the probability of finding the exit point along the fixed search direction is very low. Therefore, we adjust the direction g_i by the following gradient descent equation:

$$\mathrm{g_{i+1}} = \mathrm{g_i} - \rho_2 \cdot \nabla_{\mathrm{g_i}} \mathrm{F(w_i)} \tag{10}$$

where ρ_2 is the learning rate for the adjustment, and $\nabla_{g_i} F(w_i)$ is the gradient of the loss function $F(w, x)$ w.r.t. g_i.

When the exit point is found or ρ_1 increases to ρ_{max}, the trajectory stops moving and the local solver is called up again to find a new (Tier-1) SEP from the exit point.

Theoretical basis

The stability region

The solution of the deep neural network (1) starts from $w_e \in R^N$ at $t = 0$ is called a trajectory and denoted as $\phi(\cdot, w_0)$. Define $w_e \in R^N$ as equilibrium point of (1) if $\dot{w}|_{w_e} = 0$. An equilibrium point is a degenerate trajectory. For every $\epsilon > 0$, there is a $\delta > 0$ such that $\|w_0 - w_e\| < \delta$ implies $\|\phi(t, w_0) - w_0\| < \epsilon, t > 0$, then w_e is stable. $A(w_e)$ is defined as the stability region of SEP w_e, and in this region, all trajectory converges to the w_e.

$$A(w_e) = \left\{ w \in R^n : \lim_{t \to \infty} \varphi(t, w) = w_e \right\} \tag{11}$$

If the real part of the eigenvalue of the Jacobian matrix $\nabla F(w_e)$ is not 0, then the equilibrium point w_e is termed hyperbolic [42]. Furthermore, the real parts of the eigenvalue of $\nabla F(w_e)$ have exactly k positive, and w_e is a type-k hyperbolic equilibrium point. A type-k equilibrium point is unstable for all $k \geq 1$. Given a type-k equilibrium point w_e, its stable manifold W^s and unstable manifold W^u are defined by:

$$\begin{aligned} W^s(w_e) &= \left\{ w \in R^n : \lim_{t \to \infty} \varphi(t, w) = w_e \right\} \\ W^u(w_e) &= \left\{ w \in R^n : \lim_{t \to -\infty} \varphi(t, w) = w_e \right\} \end{aligned} \tag{12}$$

Observe that $W^s(w_e) = A(w_e)$ when w_e is a type-0 equilibrium point.

A comprehensive theoretical work in characterizing the stability region and the stability boundary has been developed [42–45]. If the quotient gradient system (5) satisfies the following assumptions, then its stability boundary can be fully characterized.

A1) All the equilibrium points on the stability boundary are hyperbolic.
A2) The stable and unstable manifolds of equilibrium points on the stability boundary satisfy the transversality condition.
A3) Every trajectory on the stability boundary approaches one of the equilibrium points as $t \to \infty$.

Remark: Assumption A1) is a general property of quotient gradient system (3) and may be verified for a specific system by directly computing the eigenvalues of the corresponding Jacobian matrix of the vector field. Assumption A2) is also a general property, but it is difficult to be check. Although assumption A3) is not a general property, it can be checked in many systems using the V-function or direct analysis.

Theorem 1 (Characterization of the Stability Boundary) [42]: *Consider a nonlinear dynamical system* (5) *that satisfies assumptions A*1) *and A*3). *Let* $w_i^e, i \geq 1$ *be the equilibrium points on the stability boundary* ∂A *of a SEP, say* w_s.

Then, the stability boundary is completely characterized as follows:

$$\partial A(w_s) = \bigcup_{i \geq 1} W^s(w_i^e). \tag{13}$$

This theorem asserts that the stability boundary of a class of nonlinear dynamical systems satisfying assumptions A1) and A3) can be completely characterized and it equals the union of the stable manifolds of the equilibrium points on the stability boundary.

We note that in solving problem (1), the following sequence of unconstrained optimization problems are solved instead [46, 47]:

$$\min_{w} F(w,x) = F(w,x) = \begin{bmatrix} h(w,x_1) \\ h(w,x_2) \\ \cdots \\ h(w,x_M) \end{bmatrix}, w \in R^N \tag{14}$$

We define the following nonlinear dynamical system:

$$\begin{aligned} \dot{w} &= -\alpha \cdot \nabla F(w)^T \cdot F(w) \\ &= -\alpha \begin{bmatrix} \nabla_w h(w,x_1) \\ \nabla_w h(w,x_2) \\ \cdots \\ \nabla_w h(w,x_M) \end{bmatrix}^T \begin{bmatrix} h(w,x_1) \\ h(w,x_2) \\ \cdots \\ h(w,x_M) \end{bmatrix}, w \in R^N \end{aligned} \tag{15}$$

Two important properties of system (15) are to be explored in the following to compute multiple LOSs of the general nonlinear optimization problem (1). These two properties are examined as follows.

Complete stability

Theorem 2 (Complete Stability) [43, *Section IV*]: *Every trajectory of quotient gradient system* (15) *converges and all converge to an equilibrium point. In addition, almost every trajectory converges to a SEP of* (15).

This theorem states that every trajectory converges to an equilibrium point, indicating that the system behavior is simple and does not allow complex trajectory behavior [45]. The trajectory must converge to an equilibrium point of (15) from an initial point.

Furthermore, every trajectory converges to SEPs except for the trajectory glow on the stable boundary, which converges to an unstable equilibrium point. In addition, we also need to prove that the trajectory of (15) converging to a SEP is equivalent to solving a LOS for problem (1). The next section determines this through the equivalence relationship between the LOSs of (1) and the SEPs of (15).

Equivalence relations

Theorem 3 (Equivalence Relations): *Consider the nonlinear optimization problem* (1), *which corresponds to the nonlinear dynamical system* (15) *and satisfies assumptions A*1) *and A*2). *Then, the following properties hold.*

1) If w^* is a local optimal solution of (1), then w^* is a stable equilibrium point of system (15).

2) If w^* is a stable equilibrium point of (15), w^* is a local optimal solution of (1).

Proof: 1) Given the w^* as a LOS, clearly $\nabla_W F(w^*, x) = 0$, by (14). For system (15):

$$\begin{aligned}\dot{w} &= -\alpha \cdot \nabla F(w)^T \cdot F(w) \\ &= -\alpha \begin{bmatrix} \nabla_w h(w, x_1) \\ \nabla_w h(w, x_2) \\ \cdots \\ \nabla_w h(w, x_M) \end{bmatrix}^T \begin{bmatrix} h(w, x_1) \\ h(w, x_2) \\ \cdots \\ h(w, x_M) \end{bmatrix} = 0, w \in R^N \end{aligned} \tag{16}$$

Since w^* is a local optimal solution, there exists a vector d $d^T \nabla^2_{ww} F(w^*, x) d > 0$ [48]. It needs to be proven that w^* is a hyperbolic SEP. Consider the Jacobian of $\nabla_w F$ in (15):

$$\mathrm{J(w,x)}{=}\nabla^2_{\mathrm{ww}}\mathrm{F(w,x)} \tag{17}$$

Next, we show that the quadratic form $J(w, x) \doteq w^T J(w^*, x), w > 0$, for $\forall (w, x) \neq 0$. Let

$$\mathrm{P(w,\ x)} = \mathrm{w^T}\ \nabla^2_{\mathrm{ww}}\mathrm{F(w^*,\ x)w} \tag{18}$$

Clearly, $J(w, x) = P(w, x)$.
By the above-verified claim, the quadratic form $J(w, x) = w^T J(w^*, x) w = P(w, x)$, which shows all $J(w^*, x) = P(w, x) > 0$ for all $w \neq 0$. For $J(w^*, x)$ is symmetric, $J(w^*, x)$ is a positive definite square matrix, and the characteristic values of $J(w^*, x)$ are all positive real numbers. Therefore, w^* is a type-0 and the hyperbolic equilibrium point of (15), due to $\nabla_w(-\nabla_w F) = -J(w^*, x)$, when $w = w^*$. So, 1) is proved.

Proof: 2) First, we claim that w^* is a feasible point of the problem (1), $w^* \in S$. Given the SEP w^* of (15), thus $\nabla_w F(w^*, x) = 0$. Then, w^* is a LOS of (1), such that w^* is a feasible point of (1).
w^* is a (hyperbolic) SEP of (15). Therefore, w^* is an isolated local minimum point of $F(w, x)$ [43]. Then, define $\Omega \subseteq R^{N+M}$ as a neighborhood of w^*, such that $F(w^*, x) < F(w, x)$, for $\forall (w, x) \subseteq \Omega$ with $w \neq w^*$. A neighborhood $U_{w^*} \subseteq R^{N+M}$ of w^* exists such that:

$$U_{w^*} \doteq \{\mathrm{w};\ \mathrm{w} \subseteq U_{w^*}\} \subseteq \Omega \tag{19}$$

Hence, there exists a neighborhood $U^{\pi}_{w^*} \subseteq U_{w^*}$ of w^*. Or, let $(w^{\pi}, x) \in U^{\pi}_{w^*} \cap S \subseteq \pi$. Then,

$$h(w^*) < h(w^{\pi}) \tag{20}$$

where $w^{\pi} \in U^{\pi}_{w^*} \cap S$, $w^{\pi} \neq w^*$. Therefore w^* is a local optimal solution for problem (1). Thus, assertion 2) is proved.

So far, the equivalence relationship between the LOSs of (1) and the SEPs of (15) has been proved. This is the key to ensuring the effectiveness of the CPSOTJUTT methodology.

CPSOTJUTT-based ensemble model

This paper develops a three-layer ensemble model of CPSOTJUTT-EM for automatically designing and training DNN architectures for power line inspection, as shown in Fig. 1. The pseudocode of CPSOTJUTT-EM is shown in Algorithm 3.

Algorithm 3 Pseudocode of the CPSOTJUTT-EM.

Initialize the population, the number of individuals is N, generation is T, define the fitness function $f(\boldsymbol{x})$, minimum is ε:
for j=1 to epoch **do**
  Select the population.
  Crossover, mutation operation.
  Calculate the fitness function $f(x)$. Search for the optimal solutions by CPSOTJUTT.
  Update the evolutionary population.
  If $\Delta f(x) < \varepsilon$ **then**
  break
end for
Output the optimal individual (sub-DNN).
Use the optimal particle to generate an ensemble model.
Train the ensemble model weights by CPSOTJUTT.

In this paper, the CPSOTJUTT-EM is developed to enhance the performance of automatically designed DNN architectures in two aspects:

1) Enhance the robustness of the DNN training method by applying the CPSOTJUTT to efficiently build multiple high-quality classification engines with different DNN architectures.
2) Improve the generalization ability through the ensemble model by applying the CPSOTJUTT methodology to build an effective ensemble of multiple members to achieve a higher accuracy and generalization ability in power line inspection.

Bottom layer: design the DNN architecture

In this layer, the genetic algorithm (GA) is used to design high-quality DNN architectures stably and quickly. Similar constructive strategies of DNNs have been widely studied and achieved satisfactory results [21]. We provide binary code representation of a DNN architecture for the GA method and automatically designed high-quality

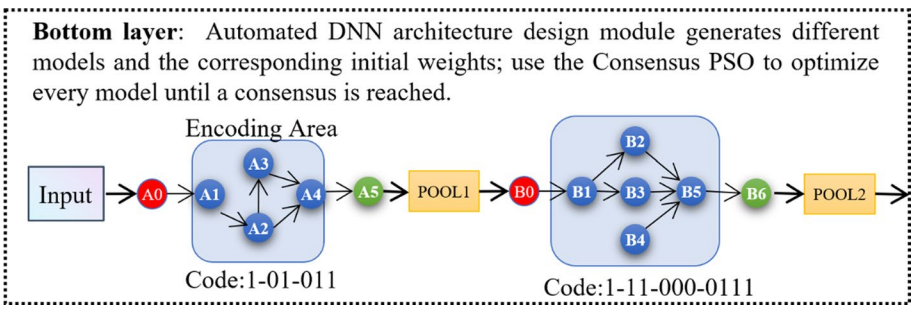

Fig. 5 Binary code representation of DNN architecture

DNN architectures to serve as the fundamental DNN for the subsequent stages. The binary code representation of the DNN architecture is shown in Fig. 5.

The encoding area begins with the second convolutional layer and represents the connection between the current and previous convolutional layers: 1 indicates that there is a connection, while 0 indicates that there is no connection. There is a fixed input node and an output node in each stage. Besides the convolutional layer, there are also batch normalization and ReLU, which are proven to play a positive role in DNN training. Fully connected parts are preset.

The model designed through automatic DNN architecture algorithm usually needs to be further customized according to the requirements of specific tasks. For classification model, it is usually necessary to add a fully connected layer at the end of the network, followed by a SoftMax layer, in order to output classification tasks. The fully connected layer is responsible for converting the feature maps extracted by the convolutional layer into the final classification results.

For object detection model, a common method is to use region proposal network (RPN) to generate candidate object regions, and then require a region of interest (RoI) pooling layer to extract fixed size feature representations from region proposals of different sizes for input into classification and regression. On the basis of the automatic DNN architecture design algorithm, the RPN layer can be added at an appropriate location to achieve object detection.

However, the method of DNN training is very sensitive to the initial points, so the true capacity of the designed DNN architecture is difficult to verify by a single training. The CPSOTJUTT methodology proposed in this paper can quickly and robustly train and select the high-quality DNN architecture designed by the GA algorithm.

Middle layer: build diverse optimal DNN classification engines

In this layer, based on the optimal DNN architecture designed from the bottom layer and the corresponding initial guess w^*, the CPSOTJUTT methodology proposed in this paper explores a set of diverse optimal DNN classification engines:

$$\min F(w)|_x \tag{21}$$

where $w*$ is the initial guess of the consensus state reached by CPSO in the bottom layer.

Stage II of the CPSOTJUTT methodology proposed in this paper quickly and robustly converges to the SEP (Tier-0) from w*. Then we use the TRUST-TECH methodology to jump out of the current region, enter the stability region of neighboring SEPs, and obtain multiple SEPs in a tier-by-tier search manner.

We apply the CPSOTJUTT methodology to train these DNN architectures to obtain high-quality DNN classification engines in the middle layer.

Top layer: the DNN-based ensemble model

In this layer, diverse high-quality DNN classification engines from the middle layer are used as the hidden nodes of the ensemble model, and the CPSOTJUTT methodology is used for training to find the weight σ. The final output of the ensemble model is as follows:

$$h(x) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} t_{ic}\log\left(\sum_{j=1}^{M}\sigma_{jc}p_{jc}\right) \tag{22}$$

where x is the input data, t_{ic} is the one-hot value of the class, σ_{jc} is the weight of the *c-th* classification of the *j-th* sub-DNN, and p_{jc} is the probability that sample i belongs to class c by the *j-th* sub-DNN, N is the size of Mini-Batch, C is the number of classes, and M is the number of sub-DNN for the ensemble model.

The structure diagram of the classification ensemble model is shown in Fig. 6. In the figure, F is different feature extractor automatically designed by the automatic architecture algorithm, FC is the fully connected layer and SM is the SoftMax layer.

The ensemble function of the object detection ensemble model adopts the weighted bounding box fusion method. Assuming that the data of the bounding box is stored in set B. B_c contains a bounding box labeled c. F_c is the ensemble result of bounding boxes in B_c, represented as $(s, x_{tl}, x_{br}, y_{tl}, y_{br})$. When the bounding box of the *jth* sub-DNN is added to B_c, the confidence level of the ensemble bounding box F_c is recalculated as:

$$F_c = \frac{\sum_{A_k \in B_c} A_k(s)}{s_1 + s_2 + \cdots + s_N} \tag{22}$$

where, N represents the total number of bounding boxes contained in B_c, and A_k is one of the bounding boxes, namely $(s, x_{tl}, x_{br}, y_{tl}, y_{br})$.

The coordinates of the ensemble bounding box are updated as follows:

$$\begin{cases} F_c(x_{tl}) = \frac{\sum_{A_k \in B_c} \sigma_{jc} A_k(s) A_k(x_{tl})}{s_1 + s_2 + \cdots + s_N} \\ F_c(x_{br}) = \frac{\sum_{A_k \in B_c} \sigma_{jc} A_k(s) A_k(x_{br})}{s_1 + s_2 + \cdots + s_N} \\ F_c(y_{tl}) = \frac{\sum_{A_k \in B_c} \sigma_{jc} A_k(s) A_k(y_{tl})}{s_1 + s_2 + \cdots + s_N} \\ F_c(y_{br}) = \frac{\sum_{A_k \in B_c} \sigma_{jc} A_k(s) A_k(y_{br})}{s_1 + s_2 + \cdots + s_N} \end{cases} \tag{23}$$

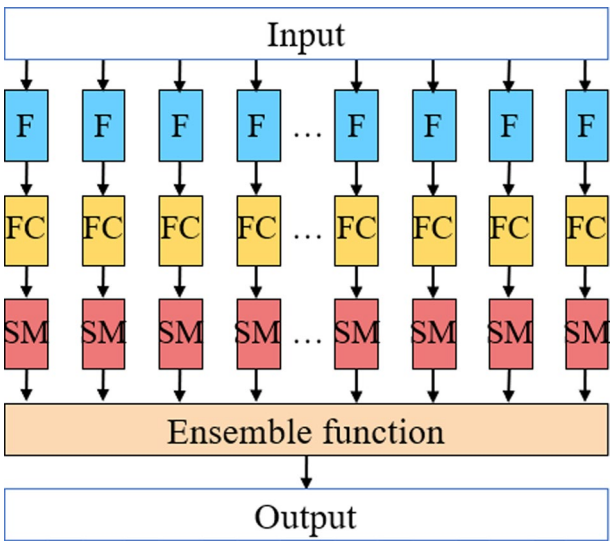

Fig. 6 Structure diagram of classification ensemble model

where, $s_2 = A_k(s)$ is the confidence value of A_k.

The structural diagram of the object detection ensemble model is shown in Fig. 7.

As shown in Fig. 7, F is the backbone network automatically designed by the automatic architecture algorithm, RPN is the Region Proposal Network (RPN), RoI is region of interest, N is the network block composed of convolutional layers, C is classification prediction, and B is boundary box prediction.

In this layer, the ensemble model composed of diverse high-quality DNN model achieves higher performance than single model in the middle layer. The ensemble model is an optimal combination of diverse high-quality DNN model, which can fully exploit the advantages of each sub-DNN, and further improve the overall accuracy and generalization capability.

The execution process of the CPSOTJUTT-EM is relatively complex. In order to clearly display the execution status of each algorithm, we have created a flowchart, as shown in Fig. 8.

Experiment

With the increasing dependence of society on electricity, ensuring the continuous power supply of the power system has become an important component of power supply guarantee. Power system inspection is a key measure to ensure the stable and safe operation of the power system [49–51]. With the rapid development of information technology, new technologies such as unmanned aerial vehicle (UAV) inspection and robot inspection have gradually replaced traditional manual inspection method [52], bringing new opportunities for power system inspection. At the same time, deep learning (DL) has made rapid progress in computer vision technology, especially in the fields of power line

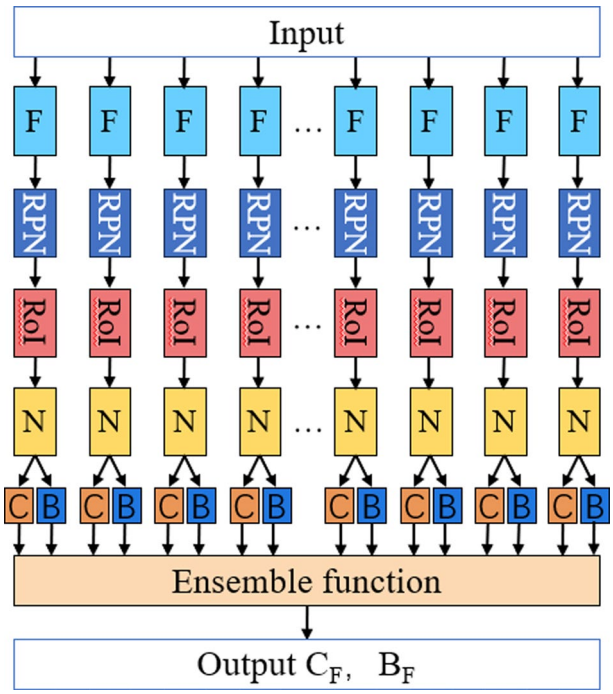

Fig. 7 Structure diagram of object detection ensemble model

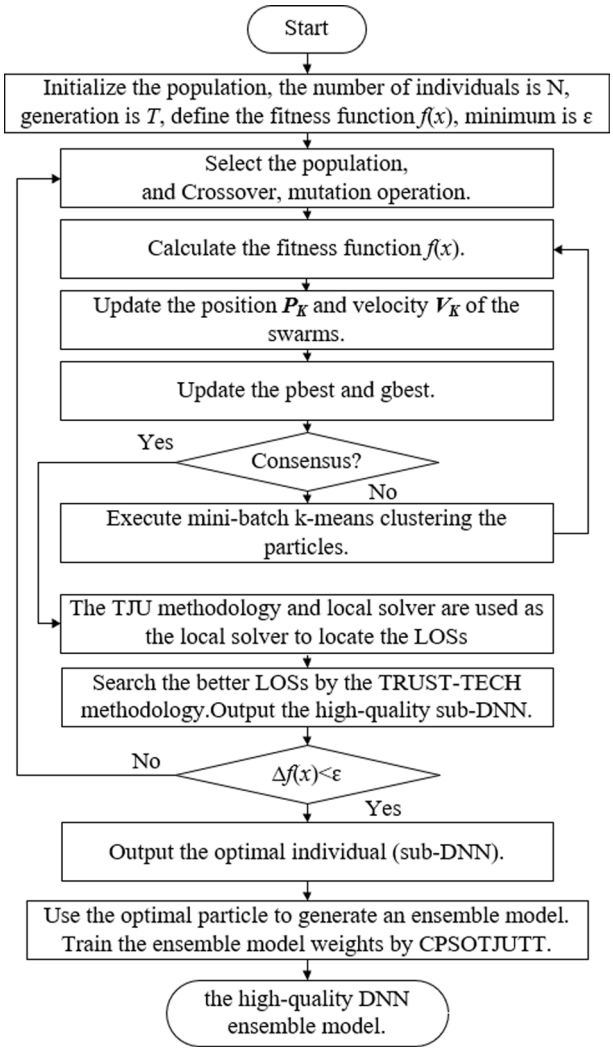

Fig. 8 The flowchart of the CPSOTJUTT-EM

object detection and defect recognition, where computer vision technology has been widely applied [53]. The application of deep learning technologies provides more efficient and accurate methods for power system inspection, further improving the safety and reliability of power system operation [54, 55]. The examples of the power system inspection object are shown in Fig. 9.

In this paper, three self-made power system inspection datasets are independently developed for the three key areas of power system inspection, these datasets still meet the requirements for classification model testing and object detection model testing:

Power Line Insulator Inspection Dataset: Power line insulator inspection dataset (PLIID) made in this study consists of 60,000 color images of insulator defects, including 4 classes of defects: ceramic insulator edge loss, ceramic insulator middle loss, glass insulator edge loss, glass insulator middle loss.

Power System Substation Inspection Dataset: Power system substation inspection dataset (PSSID) consists of images of internal equipment defects in power system

Fig. 9 Examples of the power system inspection object

substations, including 9 classes of color images: suspended matter, component oil contamination, bird nests, ground oil pollution, metal corrosion, meter inspection, oil seal damage, silicone discoloration, dial blurriness.

Power line obstacle inspection dataset: Power line obstacle inspection dataset (PLOID) is composed of images of obstacles along the power line, which contains 10 classes of color images: forklift, crane, wire foreign object, tipper, wildfires, smog, Cement pump trucks, tower cranes excavator and other construction machinery.

To evaluate the effectiveness of the proposed framework, we also conduct numerical experiments on public datasets CIFAR-10 and CIFAR-100, and discuss the proposed results.

Public dataset and server configuration

MNIST: The handwritten dataset, commonly known as MNIST, is a fundamental benchmark dataset extensively utilized in the field of machine learning and computer vision. It comprises a collection of grayscale images, each representing a handwritten digit from 0 to 9. With a total of 70,000 examples, it is divided into a training set of 60,000 images and a testing set of 10,000 images.

CIFAR: CIFAR is a picture classification dataset that includes CIFAR-10 and CIFAR-100. The CIFAR-10 dataset contains 60 k 32×32 color images divided into 10 classes, each with 6 k images. The CIFAR-100 dataset contains 60 k 32×32 color images divided into 100 classes, each with 600 images.

Server configuration: We use four servers to evaluate the model proposed in this paper: Intel CPU Core (2.67 GHz) and eight GeForce RTX 2082TI GPUs. The software

framework is Pytorch, and the operating system is Linux Ubuntu 18.04. We also use python and its CPU plug-in for a test.

Convergence verification of CPSOTJUTT methodology

We train many initial points and use the filter-normalized random direction method to draw the landscape around the LOSs, as shown in Fig. 10. To see convergence regions, the results are shown as contour plots rather than surface plots.

In the experimental setup to understand the influence of initial points on convergence, we carefully adjusted the SGD method to train the lenet-5 network with a random initial point. The dataset is the handwritten dataset MNIST, and the experiment was repeated 100 times, as shown in Fig. 11.

There are obvious boundaries according to different test accuracies of a DNN trained by the local solver and CPSOTJUTT methodology, as shown in Fig. 11. We have statistics on the number of convergence regions with various test accuracies, and the time used for SGD is calculated at 20 epochs, and CPSOTJUTT is calculated at 20 epochs after reaching or over consensus, as shown in Table 1.

Figure 11 and Table 1 show that when SGD is applied to train the DNN, 18% of the initial points converge to the optimal convergence region, and each convergence region has obvious boundaries. The CPSOTJUTT methodology has better global convergence ability, and 83% of the initial points converge to the optimal convergence region. The CPSOTJUTT methodology with an over consensus state did not achieve better results while increasing the time cost.

Test results of the CPSOTJUTT-EM on the CIFAR

To evaluate the effectiveness of our proposed CPSOTJUTT-EM model, we first tested the automatic architecture design algorithm. The experiment aims to verify the performance of the proposed automatic architecture design algorithm in image classification tasks.

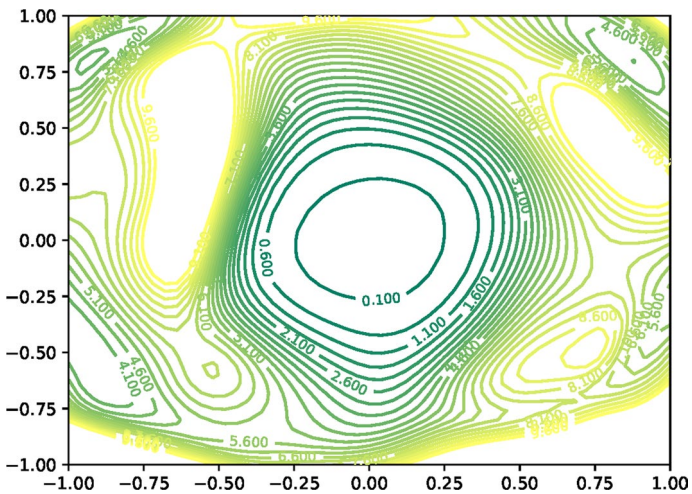

Fig. 10 2D visualization of the loss surface of DNN

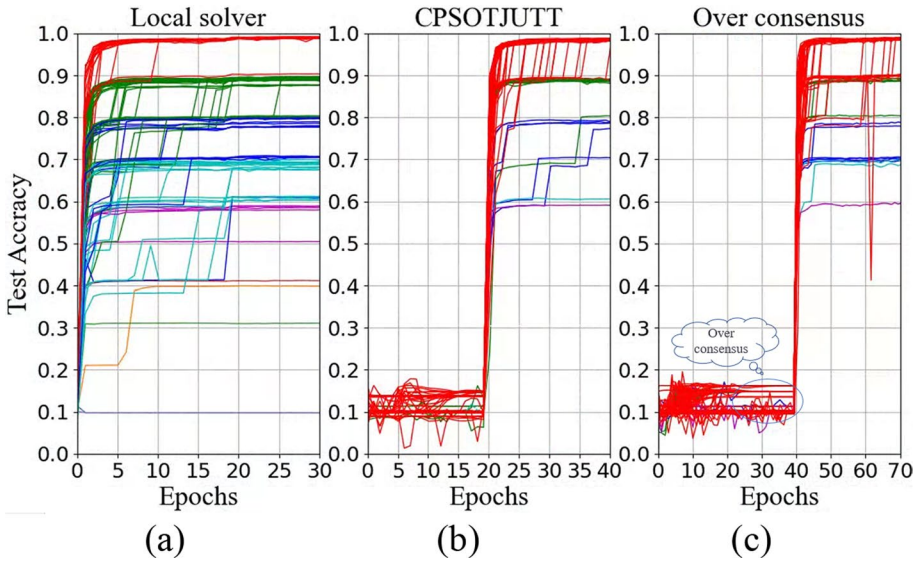

Fig. 11 Test results of a DNN trained by the local solver and the CPSOTJUTT methodology. **a** The well-tuned SGD method. **b** The CPSOTJUTT method. **c** The CPSOTJUTT methodology with over consensus

Table 1 Statistics on the number of stability regions with various test accuracies

Accuracy	< 0.65	[0.65,0.75)	[0.75,0.85)	[0.85,0.95)	[0.95,1]	Time
Frequency (Local solver)	15	19	12	36	18	17.7 h
Frequency of CPSOTJUTT	2	1	5	9	83	19.6 h
Frequency (Over consensus)	1	2	6	9	82	21.4 h

We selected some the peer competitors and divided them into three different categories. The first category covers the most advanced manually designed architectures (MD) DNN models, including ResNet [3], DenseNet [56], VGG [57]. Specifically, we used two different versions of ResNet in the experiment, namely ResNet-101 and ResNet-1202. The second category includes DNN architecture design algorithms with semi-automatic (SM) methods, such as Genetic CNN [20], Hierarchical Evolution [58], EAS [59], and Block QNN-S [60]. The third category covers methods for fully automated (FA) design methods, including large-scale evolution [18], CGP-CNN [61], NAS [59], and AE-CNN [21]. The experiment selected two widely used image classification benchmark datasets, namely CIFAR10 and CIFAR100.

To maintain fair comparison, we followed the parameter settings commonly used by the peer competitors. The population size and the number of generations are all set to 20, and the probabilities of crossover and mutation are set to 0.9 and 0.2, respectively. In addition, we set the parameters of the SGD optimizer, including momentum 0.9, learning rate 0.01 and d the learning rate is decayed by a factor of 0.0005, according to the conventions of competitors.

In this article, to evaluate the computational complexity, the indicator "GPU days" was used. The calculation method for GPU days is obtained by multiplying the number of GPU cards used by the number of days executed to find the optimal

architecture. We refer to the optimal model generated by the automatic architecture algorithm as CPGA-DNN. According to the experimental results, the performance of the proposed CPGA-DNN algorithm is superior to manually designed state-of-the-art CNN models on the CIFAR10 and CIFAR100 datasets. The results are shown in Table 2, CIFAR10 and CIFAR100 represent the test error of the model on this dataset, unit: %.

Specifically, on the CIFAR10 and CIFAR100, compared to manually designed architectures, the CPGA-DNN exhibits lower test errors. Although the parameter size of CPGA-DNN on CIFAR10 is relatively large compared to ResNet-101 and DenseNet-40, the increase in computational complexity is not significant for existing hardware devices. Compared to semi-automatic competitors, CPGA-DNN exhibits superior performance on CIFAR10 and CIFAR100, surpassing algorithms such as Genetic CNN, EAS, and Block QGS-S. Although Hierarchical Evolution slightly leads CPGA-DNN on CIFAR10, CPGA-DNN only consumes 1/14 of the GPU days required for Hierarchical Evolution. In fully automated competitors, CPGA-DNN performs best on the CIFAR10 and CIFAR100 datasets, with better test error, number of parameters, and GPU days than other methods, including Large-scale Evolution, CGP-CNN, NAS, and AE-CNN. On the CIFAR10 and CIFAR100, the test errors of CPGA-DNN were 3.67% and 16.55%, respectively. These results demonstrate the superiority and efficiency of our proposed automatic architecture design algorithm in designing DNN architectures, providing a more reliable and efficient automation method for solving image classification problems.

The above experiments show that the capability of a DNN architecture may not be fully exhibited given that local solvers may converge to bad LOSs, and thus high-quality DNN architectures are missed. We evaluate the robustness of CPSOTJUTT in automated DNN architecture design. And recorded the best network architecture (BNA), as shown in Table 3.

Table 2 Comparisons between the proposed algorithm and the state-of-the-art peer competitors

Model	CIFAR10	CIFAR100	#Parameters	GPU days	Categories
DenseNet-40 (k = 12)	5.24	24.42	1.0 M	–	MD
ResNet-101	6.43	25.16	1.7 M	–	MD
ResNet-1202	7.93	27.82	10.2 M	–	MD
VGG	6.66	28.05	20.01 M	–	MD
Genetic CNN	7.1	29.05	–	17	SM
Hierarchical Evolution	3.63	–	–	300	SM
EAS	4.23	–	23.4 M	10	SM
Block-QGS-S	4.38	20.65	6.1 M	90	SM
Large-scale Evolution	5.4	–	5.4 M	2750	FA
Large-scale Evolution	–	23	40.3	2750	FA
CGP-CNN	5.98	–	2.64 M	27	FA
NAS	6.01	–	2.5 M	22,400	FA
AE-CNN	4.3	–	2 M	27	FA
AE-CNN	–	20.85	5.4 M	36	FA
CPGA-DNN	3.76	–	1.56 M	22	FA
CPGA-DNN	–	17.35	4.67	34	FA

Table 3 Recognition test error (%) on the CIFAR-10

Gen	Local solver				CPSOTJUTT			
	Min%	Max%	Avg%	Med%	Min%	Max%	Avg%	Med%
0	14.07	18.29	15.65	15.56	3.57	7.79	5.15	5.06
1	14.14	16.12	15	14.9	3.58	5.55	4.46	4.35
2	14.09	16.07	14.72	14.62	3.52	5.52	4.20	4.12
3	14.02	16.57	14.63	14.48	3.48	6.07	4.06	3.98
5	13.81	17.40	14.69	14.43	3.21	6.89	4.13	3.93
8	13.45	15.34	14.23	14.23	2.93	4.80	3.65	3.73
10	13.32	16.08	14.34	14.26	2.67	5.46	3.65	3.62
20	13.21	15.10	13.61	13.26	2.04	3.78	2.76	2.71
30	13.1	15.69	13.65	13.56	–	–	–	–
50	12.96	14.74	13.5	13.23	–	–	–	–
BNA	1-01|0-01-100|0-11-101-0001				1-01|0-01-101|1-01-101-0111			

Table 4 Optima performances by CPSOTJUTT on CIFAR-10 dataset

Optimal index	Train error (%)	Test error (%)	Distance to Tier-0
Tier-0(ep 100)	2.54	2.85	0
Tier-0(ep 200)	2.01	2.36	0.42
Tier-1 (#1)	**1.77**	**2.13**	8.24
Tier-1 (#2)	1.90	2.17	8.20
Tier-1 (#3)	1.91	2.24	7.58
Tier-1 (#4)	1.93	2.39	1.13
Tier-1 (#5)	1.92	2.38	0.98
Tier-1 (#6)	1.99	2.21	7.95
Tier-1 (#7)	1.96	2.32	1.00
Tier-1 (#8)	1.93	2.16	7.89
Tier-1 (#9)	2.00	2.23	7.18
Tier-1 (#10)	2.05	2.32	8.00
Best Percentage Improvement	0.77%	0.72%	–

Table 3 demonstrates that the proposed CPSOTJUTT can design higher-quality DNN architecture faster than the local solver. To explain more clearly each stage of the CPSOTJUTT, we design the following experiment:

Step 1: Converge quickly and robustly to the SEP (Tier-0) using stage II of the CPSOTJUTT methodology.
Step 2: Search for exit points on 10 paths, starting from the Tier-0 using the TRUST-TECH methodology.
Step 3: Converge quickly and robustly to the SEP (Tier-1) from each exit point using stage II of the CPSOTJUTT methodology.

The experimental results of the CPSOTJUTT are given in Table 4. The DNN architecture is the BNA selected in the first layer of the CPSOTJUTT-EM.

Table 5 Optima performances by CPSOTJUTT (Tier-2, starting from the best tier-1 LOS (#1, according to the test error)) on the CIFAR-10 dataset

Optimal index	Train error (%)	Test error (%)	Distance to Tier-0
Tier-1(#1)	1.77	2.13	0
Tier-0(ep 300)	1.46	2.1	7.3686
Tier-2 (#1)	1.28	1.85	7.8153
Tier-2 (#2)	1.23	1.87	8.3939
Tier-2 (#3)	1.20	**1.71**	9.0559
Tier-2 (#4)	1.17	1.79	6.5511
Tier-2 (#5)	**1.16**	1.77	8.2235
Tier-2 (#6)	1.26	1.75	8.8999
Tier-2 (#7)	1.19	1.95	10.0292
Tier-2 (#8)	1.17	1.83	7.7495
Tier-2 (#9)	1.24	1.74	9.2371
Tier-2 (#10)	1.22	1.89	8.4207

Table 6 Test results of CPSOTJUTT (Tier-1) on CIFAR-100

Optimal index	CIFAR-100	
	Test error (%)	Distance to Tier-0
Tier-0	9.70	0
Tier-1 (#1)	6.98	37.2742
Tier-1 (#2)	5.99	36.0624
Tier-1 (#3)	6.90	45.4467
Tier-1 (#4)	6.01	40.8781
Tier-1 (#5)	6.27	38.1602
Tier-1 (#6)	5.98	35.5248
Tier-1 (#7)	6.14	43.4156
Tier-1 (#8)	6.43	44.1228
Tier-1 (#9)	6.42	34.0974
Tier-1 (#10)	6.95	37.9917
Tier-1 (average)	6.25	–

Table 4 shows 10 search paths of Tier-1 by the CPSOTJUTT. The results show that a LOS better than Tier-0 can be obtained on each search path, and the best one (Tier-1 (#1)) reduces the test error by 0.72% compared to Tier-0 (ep 100). Tier-0 is trained for 200 epochs, and the majority of the Tier-1 train error and test error are lower than for Tier-0 (ep 200). The findings suggest that searching for greater quality LOSs at Tier-0 with additional iterations is challenging.

To expand the search space, we further search Tier-2 starting with the best point of Tier-1(#1), as shown in Table 5.

As shown in Table 5, all Tier-2 solutions outperform Tier-1 with limited incremental time cost. The results show that the TRUST-TECH methodology can efficiently explore high-quality LOSs in the parameter space.

To evaluate the performance of CPSOTJUTT on a large dataset, the CIFAR-100 dataset was used, and the results are given in Table 6, indicating the competitive capability of CPSOTJUTT.

Table 7 The proportion of all classes of PLOID

Obstacle	Label	Number	Proportion (%)
Forklift	Forklift	5228	6.34
Crane	Crane	30,617	38.27
Foreign body	DXYW	12,885	16.1
Tipper	Tipper	4476	5.59
Other machinery	QTSGJX	3076	3.85
Wildfire	Wildfire	2597	3.25
Cement pump truck	CPT	2053	2.57
Tower Crane	TowerCrane	9780	12.23
Excavator	Excavator	7541	9.42
Smog	Smog	1747	2.18

Table 8 Testing results of CPSOTJUTT (Tier-1) on PLOID

Optimal index	PLOID	
	Test error (%)	Distance to Tier-0
Tier-0	3.17	7.6978
Tier-1 (#1)	3.21	7.7448
Tier-1 (#2)	3.21	7.3625
Tier-1 (#3)	3.39	3.5845
Tier-1 (#4)	3.37	3.5722
Tier-1 (#5)	3.21	7.5635
Tier-1 (#6)	3.37	3.5382
Tier-1 (#7)	**3.10**	7.6426
Tier-1 (#8)	3.23	7.2184
Tier-1 (#9)	3.33	7.7385
Tier-1 (#10)	3.56	7.3570
Tier-1 (average)	3.30	–

Bold value indicates the lowest test error

The above experiments show that CPSOTJUTT-EM achieves competitive results in testing on the CIFAR dataset. Further evaluations of the performance of CPSOTJUTT-EM are given in later sections.

Testing results of the CPSOTJUTT on imbalanced PLOID dataset

We evaluate the ability of the CPSOTJUTT methodology to effectively handle imbalanced datasets in a real-world application for drone-based visual inspection of electric power transmission line corridors. Table 7 shows the results of a statistical analysis of the PLOID, which has a huge imbalance with the proportion of the largest and smallest classes being 38% and 2%, respectively. The imbalance of the PLOID dataset makes the classification task a big challenge.

As shown in Tables 8 and 9, we compared CPSOTJUTT with the most used SGD method: The weight decay and momentum of SGD are fixed as 0.0001 and 0.8, respectively.

Tables 8 and 9 shows the additional research on a local solver (SGD) and CPSOTJUTT in a different model. The experimental data show that CPSOTJUTT can quickly converge to a high-quality LOS. In particular, the performance of the CPSOTJUTT is better,

Table 9 Optima performances by CPSOTJUTT on PLOID

MODEL	Local Solver	CPSOTJUTT
	Epochs Err (%)	Epochs Err (%)
BNA-1	20 9.45	14 6.15 (— 3.30)
BNA-2	20 6.24	15 4.03 (— 2.21)

Table 10 The error rates of all classes of two training methods

Obstacle	Error (%) (ResNet-50)	
	Local Solver (SGD)	CPSOTJUTT
Forklift	9.4	7.5
Crane	2.2	1.2
DXYW	5.4	4.5
Tipper	9.8	6.3
QTSGJX	10.6	8.3
Wildfire	13.1	6.1
CPT	14.9	11.5
Tower Crane	6.9	1.1
Excavator	8.6	
Smog	20.1	6.5

and the minimum error rate of the CPSOTJUTT is only 4.03%, which is 2.21% lower than that of the local solver for BNA-2. In order to test the optimization performance of CPSOTJUTT under different types of models, we chose ResNet-50 to test the test error on imbalanced datasets PLOID, as shown in Table 10.

Table 10 well demonstrates that the overall error rate of the CPSOTJUTT is lower than that of the local solver. The test error of CPSOTJUTT has a significant decline in the hard examples. In particular, the error rate of smog is 13.6% lower than the local solver. In general, CPSOTJUTT performs better performance in training DNN, especially in the case of an imbalanced dataset, which can greatly improve the accuracy of hard examples.

Performance test of classification models on three power system datasets

The proposed CPSOTJUTT-EM designs diverse high-quality DNN architectures and corresponding weights in the CIFAR and PILD datasets in the previous sections. In this section, we apply the CPSOTJUTT methodology to train the ensemble model of the above various DNN architectures, as shown in Table 11, the table shows the test error of the model on the corresponding dataset, unit: %.

We set up a comparative experiment using the DEns-VGG19 ensemble model and the proposed CPSOTJUTT-EM model. The experimental results show that the CPSOTJUTT methodology achieves a lower test error than the local solver in the training of DEns-VGG19, as well as a lower test error than the DEns-VGG19 model. The CPSOTJUTT-EM achieved lower test error in classification testing on both public datasets and three self-made power system inspection datasets.

Table 11 Evaluation of the ensemble model. DEns-NN: NN + Ensemble layer (the optimization method)

	DEns-VGG19 (Local solver)	DEns-VGG19 (CPSOTJUTT)	CPSOTJUTT-EM
CIFAR-10	5.09	3.25	0.91
CIFAR-100	18.04	11.4	5.98
PSIID	12.37	8.69	2.43
PSSID	17.50	12.68	3.56
PLOID	15.78	10.22	3.18

Table 12 Test results for different object detection model

Model	PSIID		PSSID		PLOID		#Parameters
	mAP	mAR	mAP	mAR	mAP	mAR	
YOLOv5	0.554	0.590	0.574	0.607	0.565	0.593	42.6 M
Faster R-CNN	0.656	0.702	0.633	0.658	0.621	0.672	40.23 M
Faster-FPN	0.710	0.803	0.717	0.765	0.735	0.779	36.25 M
Cascade R-CNN	0.793	0.868	0.802	0.832	0.837	0.871	256 M
CPGA-DNN	0.763	0.833	–	–	–	–	20.75 M
CPSOTJUTT-EM	0.847	0.897	–	–	–	–	132.6 M
CPGA-DNN	–	–	0.742	0.802	–	–	21.62 M
CPSOTJUTT-EM	–	–	0.843	0.879	–	–	139.7 M
CPGA-DNN	–	–	–	–	0.827	0.856	20.67 M
CPSOTJUTT-EM	–	–	–	–	0.892	0.917	129.5 M

Performance testing of object detection models on three power system datasets

In this section, we conduct performance tests on object detection model based on three self-made power system datasets: PSIID, PSSID, and PLOID. We chose classic object detection models, including YOLO v5, Faster R-CNN, Faster FPN, and Cascade R-CNN as benchmarks for comparison. We used the CPSOTJUTT-EM algorithm to generate an ensemble object detection model and selected the single model with the best accuracy from it, which is denoted as CPGA-DNN. The optimization of the model adopts the CPSOTIUTT three-stage optimization algorithm proposed in this article. The experimental results are shown in Table 12.

From Table 12, it can be observed that the CPGA-DNN outperforms models such as YOLOv5, Faster R-CNN, and Faster FPN in object detection accuracy with only 20–22 M parameters. Although there is a slight difference compared to the Cascade R-CNN, the parameter quantity of CPGA-DNN is much lower than that of Cascade R-CNN. The accuracy of the ensemble model generated by CPSOTJUTT-EM has significantly improved compared to Cascade R-CNN and single model CPGA-DNN.

Conclusion

In this paper, a novel three-layer ensemble model (CPSOTJUTT-EM) for power line inspection is developed, which can automatically design DNN architectures quickly and stably without any DNN expertise. In the CIFAR and PLOID datasets, the test errors

are reduced by 4.18%, 12.06%, and 12.6%, respectively, especially for hard examples in the PLOID dataset. The CPSOTJUTT methodology proposed in this paper has a strong global convergence ability: 83% of the initial points converge to the optimal convergence region, thereby improving the stability by 65%. The ensemble classification model and ensemble object detection model automatically generated by CPSOTJUTT-EM have achieved good results in PSIID, PSSID, and PLOID, indicating that the CPSOTJUTT-EM three-layer model can achieve high inspection accuracy in power system inspections.

In conclusion, the CPSOTJUTT-EM proposed in this paper can automatically design a high-quality ensemble model of DNN for power system inspection.

Acknowledgements
I would like to thank Mr. Chiang for his guidance on my research. His guidance and encouragement prompted me to write the whole work and complete it successfully. In the process of writing, he always made me have confidence in myself and guided me to many important publications that were quite helpful.

Author contributions
X-lL: Conceptualization, Methodology, Software, Formal analysis, Writing—original draft. H-DC: Writing—review & editing, Project administration, Supervision.

Funding
Funding information is not available.

Availability of data and materials
No new data were generated or analyzed in support of this research.

Declarations

Ethics approval and consent to participate
This article does not involve animal or human experiments, and no ethics approval is required. And written informed consent was obtained from all participants.

Consent for publication
All authors gave their consent for publication.

Competing interests
The authors declare that they have no competing interests.

Received: 3 March 2023 Accepted: 17 September 2023
Published online: 28 September 2023

References

1. Yang R, Zha X, Liu K, Xu S. A CNN model embedded with local feature knowledge and its application to time-varying signal classification. Neural Netw. 2021;142:564–72.
2. Chen T, Wang N, Wang R, Zhao H, Zhang G. One-stage CNN detector-based benthonic organisms detection with limited training dataset. Neural Netw. 2021;144:247–59.
3. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
4. Tan M, Le Q. Efficientnet: rethinking model scaling for convolutional neural networks. International Conference on Machine Learning. PMLR, 2019.
5. Sun Y, Xue B, Zhang M, Yen GG, Lv J. Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE Trans Cybern. 2020;50(99):1–15.
6. Stanley KO, Clune J, Lehman J, Miikkulainen R. Designing neural networks through neuroevolution. Nat Mach Intell. 2019;1(1):24–35.
7. Zheng Z, Li X. A novel vehicle lateral positioning methodology based on the integrated deep neural network. Expert Syst Appl. 2020;142: 112991.
8. Ahmed S, Razib M, Alam MS, Alam MS, Huda MN. Ensemble approach for improving generalization ability of neural networks. 2013 International Conference on Informatics, Electronics and Vision (ICIEV). IEEE, 2013.
9. Ganaie MA, Hu M, Malik AK, Tanveer M. Ensemble deep learning: a review. Eng Appl Artif Intell. 2022;115: 105151.
10. Chaudhari P, Choromanska A, Soatto S, LeCun Y, Baldassi C, Borgs C, Chayes J, Sagun L, Zecchina R. Entropy-SGD: biasing gradient descent into wide valleys. J Stat Mech Theory Exp. 2019;2019(12): 124018.
11. Cheridito P, Jentzen A, Rossmannek F. Non-convergence of stochastic gradient descent in the training of deep neural networks. J Complex. 2021;64: 101540.
12. Yuan K, Ying B, Sayed AH. On the influence of momentum acceleration on online learning. J Mach Learn Res. 2016;17(1):6602–67.

13. Arjevani Y, Carmon Y, Duchi JC, Foster DJ. Lower bounds for non-convex stochastic optimization. Math Program. 2022;199:165.
14. Wilson AC, Roelofs R, Stern M. The marginal value of adaptive gradient methods in machine learning. Adv Neural Inf Proc Syst. 2017; 30.
15. Luo L, Xiong Y, Liu Y, Sun X. Adaptive gradient methods with dynamic bound of learning rate. arXiv preprint arXiv:1902.09843, 2019.
16. Stanley KO, Miikkulainen R. Evolving neural networks through augmenting topologies. Evol Comput. 2002;10(2):99–127.
17. Miikkulainen R, Liang J, Meyerson E, Rawal A, Fink D, Francon O, Raju B, Shahrzad H, Navruzyan A, Duffy N, Hodjat B. Evolving deep neural networks. in Artificial Intelligence in the Age of Neural Networks and Brain Computing. Elsevier, 2019, pp. 293–312.
18. Real E, Moore S, Selle A, Saxena S, Suematsu YL, Tan J, Le QV, Kurakin A. Large-scale evolution of image classifiers. in International Conference on Machine Learning. PMLR, 2017, pp. 2902–2911.
19. Real E, Aggarwal A, Huang Y, Le QV. Regularized evolution for image classifier architecture search. Proc AAAI Conf Artif Intell. 2019;33(01):4780–9.
20. Xie L, Yuille A. Genetic CNN. in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1379–1388.
21. Sun Y, Xue B, Zhang M, Yen GG. Completely automated cnn architecture design based on blocks. IEEE Trans Neural Netw Learn Syst. 2019;31(4):1242–54.
22. Kumar A, Yin B, Shaikh AM, Ali M, Wei W. CorrNet: pearson correlation-based pruning for efficient convolutional neural networks. Int J Mach Learn Cybern. 2022;13(12):3773–83.
23. Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res. 2011; 12(7).
24. Tieleman T, Hinton G. Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. COURSERA Neural Netw Mach Learn. 2012;4(2):26–31.
25. Kingma DP, Ba J. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
26. Reddi SJ, Kale S, Kumar S. On the convergence of adam and beyond. arXiv preprint arXiv:1904.09237. 2019.
27. Yang J, Zeng X, Zhong S, Wu S. Effective neural network ensemble approach for improving generalization performance. IEEE Trans Neural Netw Learn Syst. 2013;24(6):878–87.
28. Zhang S, Liu M, Yan J. The diversified ensemble neural network. Adv Neural Inf Process Syst. 2020;33:16001–11.
29. Zhang YF, Chiang HD. Enhanced elite-load: a novel CPSOATT methodology constructing short-term load forecasting model for industrial applications. IEEE Trans Industr Inf. 2019;16(4):2325–34.
30. Turkoglu M, Yanikolu B, Hanbay D. Plantdiseasenet: convolutional neural network ensemble for plant disease and pest detection. Signal Image and Video Processing. 2021;(9): 1–9.
31. Wang Y, Wang J, Gao F, Hu P, Xu L, Zhang J, Yu Y, Xue J, Li J. Detection and recognition for fault insulator based on deep learning. 2018 11th International Congress on Image and Signal Processing, Biomedical Engineering and Informatics (CISP-BMEI). IEEE, 2018.
32. Dai G, Yuan Y, Huang W, Liu Q, Ju C. Unattended substation inspection algorithm based on improved YOLOv5. 2022 IEEE International Conference on Real-time Computing and Robotics (RCAR). IEEE, 2022.
33. Zhang W, Liu X, Yuan J, Xu L, Sun H, Zhou J. RCNN-based foreign object detection for securing power transmission lines (RCNN4SPTL). Procedia Comput Sci. 2019;147:331–7.
34. Zhang J, Zhao Y, Shone F, Li Z, Frangi AF, Xie SQ, Zhang ZQ. Physics-informed deep learning for musculoskeletal modeling: predicting muscle forces and joint kinematics from surface EMG. IEEE Trans Neural Syst Rehabil Eng. 2022;31:484–93.
35. Zhang J, Li Y, Xiao W, Zhang Z. Non-iterative and fast deep learning: Multilayer extreme learning machines. J Franklin Inst. 2020;357(13):8925–55.
36. Li S, Tan M, Tsang IW, Kwok JT-Y. A hybrid PSO-BFGS strategy for global optimization of multimodal functions. IEEE Trans Syst Man Cybern Part B (Cybernetics). 2011;41(4):1003–14.
37. Houssein EH, Gad AG, Hussain K, Suganthan PN. Major advances in particle swarm optimization: theory, analysis, and application. Swarm Evol Comput. 2021;63: 100868.
38. Sculley D. Web-scale k-means clustering. in Proceedings of the 19th International Conference on World Wide Web, 2010, pp. 1177–1178.
39. Lin T-Y, Goyal P, Girshick R, He K, Dollar P. Focal loss for dense object detection. in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2980–2988.
40. Zhu M, Nazareth JL, Wolkowicz H. The quasi-cauchy relation and diagonal updating. SIAM J Optim. 1999;9(4):1192–204.
41. Hao Z, Chiang HD, Wang B. Trust-tech-based systematic search for multiple local optima in deep neural nets. IEEE Transactions on Neural Networks and Learning Systems, pp. 1–11, 2021.
42. Chiang HD, Hirsch MW, Wu FF. Stability regions of nonlinear autonomous dynamical systems. IEEE Trans Autom Control. 1988;33(1):16–27.
43. Chiang HD, Chu CC. A systematic search method for obtaining multiple local optimal solutions of nonlinear programming problems. Circ Syst I Fundamental Theory Appl IEEE Trans-actions on. 1993;43(2):99–109.
44. Chiang HD, Alberto LFC. Stability regions of nonlinear dynamical systems: theory, estimation, and applications. Cambridge University Press; 2015.
45. Deng JJ, Chiang HD, Zhao TQ. Newton method and trajectory-based method for solving power flow problems: nonlinear studies. Int J Bifurcation Chaos. 2015;25(6):591–484.
46. Pillo GD, Grippo L. A new class of augmented lagrangians in nonlinear programming. SIAM J Control Optim. 2006;17(5):618–28.
47. Du X, Zhang L, Gao Y. A class of augmented lagrangians for equality constraints in nonlinear programming problems. Appl Math Comput. 2006;172(1):644–63.

48. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inf Proc Syst. 2012; 25.
49. Wang W, Peng W, Tong L, Tan X, Xin T. Study on sustainable development of power transmission system under ice disaster based on a new security early warning model. J Clean Prod. 2019;228:175–84.
50. Glavic M. (Deep) Reinforcement learning for electric power system control and related problems: a short review and perspectives. Annu Rev Control. 2019;48:22–35.
51. Qin X, Su Q, Huang SH. Extended warranty strategies for online shopping supply chain with competing suppliers considering component reliability. J Syst Sci Syst Eng. 2017;26(6):753–73.
52. Santos T, Moreira M, Almeida J, Dias A, Martins A, Dinis J, Formiga J, Silva E. Plined: Vision-based power lines detection for unmanned aerial vehicles. in 2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). IEEE, 2017, pp. 253–259.
53. Lan M, Zhang Y, Zhang L, Du B. Defect detection from uav images based on region-based cnns. in 2018 IEEE International Conference on Data Mining Workshops (ICDMW). IEEE, 2018, pp. 385–390.
54. Wang D, Zhao G, Chen H, Liu Z, Deng L, Li G. Nonlinear tensor train format for deep neural network compression. Neural Netw. 2021;144:320–33.
55. Aldahdooh A, Hamidouche W, Fezza SA. Adversarial example detection for DNN models: a review and experimental comparison. Artif Intell Rev. 2022;55:4403.
56. Huang G, Liu Z, Maaten LVD, Weinberger KQ. Densely connected convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.
57. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014.
58. Liu H, Simonyan K, Vinyals O, Fernando C. Hierarchical representations for efficient architecture searc. arXiv preprint arXiv:1711.00436. 2017.
59. Zoph B, Le QV. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578. 2016.
60. Zhong Z, Yan J, Wu W, Shao J. Practical block-wise neural network architecture generation. Proceedings of the IEEE Conference on Computer Vision and Pattern recognition. 2018.
61. Suganuma M, Shirakawa S, Nagao T. A genetic programming approach to designing convolutional neural network architectures. Proceedings of the Genetic and Evolutionary Computation Conference. 2017.

Publisher's Note

Xian-Long Lv received the B.S. degree and the M.S. degree in electrical engineering from the University of Jinan, Jinan, China, in 2015 and 2018, respectively. He is currently pursuing the Ph.D. degree in electrical engineering, the School of Electrical Engineering and Automation, Tianjin University, Tianjin, China. His research interests include nonlinear systems theory and global optimization methods and their applications to machine learning, power systems, and computer vision.

Hsiao-Dong Chiang (M'87–SM'91–F'97) is a professor in the School of Electrical and Computer Engineering at Cornell University in Ithaca, NY, USA. His research interests include nonlinear system theory, nonlinear computation and optimization, and their practical applications to electric power systems and to energy management systems. He and his research team have published more than 400 papers in refereed journals and conference proceedings. His publications earned an h-index of 58. He holds 18 U.S. and overseas patents and several consultant positions. He is the author of two books: Direct Methods for Stability Analysis of Electric Power Systems: Theoretical Foundations, BCU Methodologies, and Applications (Hoboken, NJ, USA: Wiley, 2011) and (with Luis F. C. Alberto) Stability Regions of Nonlinear Dynamical Systems: Theory, Estimation, and Applications (Cambridge, UK: Cambridge Univ. Press, 2015). He has served as an associate editor for several IEEE transactions and IEEE journals and as a board member for IEE Japan and is the founder of Bigwood Systems, Inc. and Global Optimal Technology, Inc. in Ithaca, NY, USA.

Na Dong received her Ph.D. degree in control theory and control application at Nankai University in 2011. She is currently an associate professor in the School of Electrical and Information Engineering, Tianjin University, China. Her current research areas encompass intelligent control algorithms, heuristic optimization algorithm, neural networks, data-driven control, deep learning, and image processing.