Neural Architecture Search using Particle Swarm and Ant Colony Optimization

Séamus Lankford[1] and Diarmuid Grimes[2]

[1] Adapt Centre, Dublin City University, Ireland
seamus.lankford@adaptcentre.ie
[2] Cork Institute of Technology, Ireland
diarmuid.grimes@cit.ie

Abstract. Neural network models have a number of hyperparameters that must be chosen along with their architecture. This can be a heavy burden on a novice user, choosing which architecture and what values to assign to parameters. In most cases, default hyperparameters and architectures are used. Significant improvements to model accuracy can be achieved through the evaluation of multiple architectures. A process known as Neural Architecture Search (NAS) may be applied to automatically evaluate a large number of such architectures.
A system integrating open source tools for Neural Architecture Search (OpenNAS), in the classification of images, has been developed as part of this research. OpenNAS takes any dataset of grayscale, or RBG images, and generates Convolutional Neural Network (CNN) architectures based on a range of metaheuristics using either an AutoKeras, a transfer learning or a Swarm Intelligence (SI) approach.
Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) are used as the SI algorithms. Furthermore, models developed through such metaheuristics may be combined using stacking ensembles.
In the context of this paper, we focus on training and optimizing CNNs using the Swarm Intelligence (SI) components of OpenNAS. Two major types of SI algorithms, namely PSO and ACO, are compared to see which is more effective in generating higher model accuracies. It is shown, with our experimental design, that the PSO algorithm performs better than ACO. The performance improvement of PSO is most notable with a more complex dataset. As a baseline, the performance of fine-tuned pre-trained models is also evaluated.

Keywords: AutoML · NAS · Swarm Intelligence · PSO · ACO · CNN

1 Introduction

The area of Auto Machine Learning (AutoML) [1] is a growing area of interest in recent years. This is reflected in the development of several open source AutoML libraries among which include Auto-WEKA [2], Hyperopt-Sklearn [3], AutoKeras [4], Auto-Sklearn [5, 6] and TPOT [7].

Despite a renewal of interest in AutoML, many of these open source solutions focus on creating simpler neural architectures. Libraries which concentrate on

generating more complex architectures, such as CNNs, are at early stages of development. Consequently they are poorly documented and often unreliable [4]. In addition, the alternative of using commercial platforms is expensive and therefore users are left with few practical or viable options.

The development of OpenNAS integrates several metaheuristic approaches in a single application used for the neural architecture search of more complex neural architectures such as convolutional neural networks. Furthermore, the effectiveness of NAS in generating good neural architectures for image classification is evaluated. Standard approaches to NAS, using the AutoKeras framework, are also incorporated into the system design.

A key aspect of the study is to contrast Swarm Intelligence (SI) algorithms for NAS. Consequently, Particle Swarm Optimization (PSO) [8] and Ant Colony Optimization (ACO) [9] have been chosen as metaheuristics for creating high performing CNN architectures for grayscale and RGB image datasets.

2 Background

2.1 Convolutional Neural Networks

CNNs are feed-forward Deep Neural Networks (DNNs) used for image recognition. The original CNN architecture was proposed by LeCun [10] and consisted of two convolution layers, two pooling layers, two fully connected (FC) layers and an output layer. Subsequently, numerous models were developed including popular ones such as ResNet [11] and VGG [12]. In this study, custom CNN architectures are created by using SI heuristics to find better combinations of convolutional, pooling and FC layers.

2.2 Auto ML

AutoML involves the automation of the entire machine learning pipeline including data augmentation, feature engineering, model selection, choice of hyper parameters and finally neural architecture selection and creation. By constrast, NAS has a more narrow focus in that it concentrates on neural architecture selection and creation [13].

Tree-based Pipeline Optimization Tool (TPOT) is an open source python package that uses genetic programming in optimizing the machine learning pipeline [7]. The library performs well on simple NAS tasks involving the scikit-learn API. Given this study involves generating more complex CNNs, rather than developing optimal pipelines, it was decided not to use TPOT as part of the initial solution architecture. However, as part of future work, it may have a role in optimizing hyper parameter selection.

AutoKeras [4] is an open source AutoML system using Bayesian optimization and network morphism for efficient neural architecture search.

2.3 Neural Architecture Search

Neural architecture search is the process of automatically finding and tuning DNNs. It has been shown that DNNs have made remarkable progress in solving many real world problems such as image recognition, speech recognition and machine translation [14]. In general, NAS systems consist of three main components: a search space, a search algorithm and an evaluation strategy. The search space sets out which architectures can be used in principle whereas the search strategy outlines how the search space is explored. Finally the evaluation strategy determines which architectures yield the best results on unseen data.

A basic approach to NAS is the brute force training and evaluation of all possible model combinations. On completion, the best performing model is selected. However, this is impractical due to the combinatorics of the problem. Using metaheuristics, such as swarm intelligence, is an alternative which seeks the best model within reasonable time constraints.

2.4 Swarm Intelligence

Swarm Intelligence, a category of Evolutionary Computing, has been used for classification problems in the following forms: Particle Swarm Optimization (PSO) [15, 16] and Ant Colony Optimization (ACO) [17].

Particle Swarm Optimization PSO belongs to the class of swarm intelligence techniques and is a population-based stochastic technique for solving optimization problems developed in 1995. An open source python library, for CNN optimization using the PSO algorithm was developed by Fernandes et al [18]. The results demonstrate that their approach, psoCNN, quickly finds CNN architectures which offer competitive performance for any given dataset.

Ant Colony Optimization ACO, modelled on the activities of real ant colonies, involves moving through a parameter space of all potential solutions to find the optimal weights for a neural network.

Using ACO, a system known as DeepSwarm was developed by Byla and Pang [19] to find high performing neural architectures for CNNs. They showed that it offers competitive performance when tested on well-known image datasets.

3 Approach

With artificial neural networks, there are many parameters to choose from such as the number of hidden network layers, number of neurons per layer, type of activation function, choice of optimizer and so on. The final network design often depends on the problem domain and is typically achieved in a time consuming trial and error fashion.

Similar problems exist with CNNs but these problems are exacerbated by the length of time, and amount of computational resources required to train

such networks. Clearly, a core objective of NAS is to find good network performance within acceptable time limits through the reduction of both the number of networks tested and the length of time required for their evaluation. The implementation of NAS can be achieved through a variety of approaches including transfer learning using pre-trained networks, network morphism or swarm intelligence. Using these approaches as its pillars, a NAS system (OpenNAS) has been built which tackles such problems [3]. OpenNAS does not enforce a particular architecture but rather it allows novel and interesting architectures to be discovered.

In this work we focus on the swarm intelligence component of the OpenNAS system. The swarm optimization techniques currently used are Particle Swarm Optimization and Ant Colony Optimization.

Table 1. Parameters for Particle Swarm Optimization

	Config A	**Config B**
Swarm		
Number of iterations	10	20
Swarm size	20	10
Cg	0.5	0.5
CNN architecture		
Minimum outputs from a Conv layer	3	3
Maximum outputs from a Conv layer	256	256
Maximum neurons in a FC layer	300	300
Minimum size of a Conv kernel	3 x 3	3 x 3
Maximum size of a Conv kernel	7 x 7	7 x 7
Minimum layers	3	3
Maximum layers	20	20
CNN Training		
# epochs for particle evaluation	5	5
# epochs for global best	100	100
Dropout rate	0.5	0.5
Batch normalize layer outputs	Yes	Yes
Probability Settings		
probability_convolution	0.6	0.6
probability_pooling	0.3	0.3
probability_fully_connected	0.1	0.1

The PSO algorithm determines how the principal CNN layer types, and their associated hyperparameters, are connected together. The generated models consist of architectures using a mix of convolutional, average pooling, max pooling and fully connected layers. In addition, dropout layers and batch normalization layers are also added to alleviate overfitting. The hyperparameters associated with each layer type are indicated in Table 1.

[3] https://github.com/seamusl/OpenNAS-v1

Particle architectures, i.e. model architectures, are compiled for a number of epochs and evaluation is carried out using the standard loss function of cross-entropy loss. Particle architectures with the smallest loss are selected by the algorithm. The number of epochs parameter for pBest must be carefully chosen since it is the main driver of both run time and model accuracy.

Using an ACO approach, the parameters used for model training in the exploration process are highlighted in Table 2. Two test configurations are considered. In the first case, 8 ants are used with 30 epochs and in the second case, 16 ants are used with 15 epochs. The depth parameter was fixed at 20.

Table 2. Parameters for Ant Colony Optimization

	Config A	**Config B**
Ant Colony		
Number of Ants	8	16
Number of Epochs	30	15
Search Depth	20	20
CNN architecture		
Kernel Sizes	1, 3, 5	1, 3, 5
Minimum layers	1	1
Maximum layers	20	20
CNN Training		
Dropout rate	0.1, 0.3, 0.5	0.1, 0.3, 0.5
Batch normalize layer outputs	Yes	Yes
Probability Settings		
pheromone start, decay, evaporation	0.1	0.1
greediness	0.5	0.5

Fine tuning was implemented by initially removing the fully connected layers from the top of the model. Two blocks are then added, each of which has a fully connected layer, a batch normalization layer and a dropout layer. The hybrid structure is then trained with the new dataset. Fine tuning of a VGG16 network is illustrated in Figure 1.

Fig. 1. Tuned VGG16 model

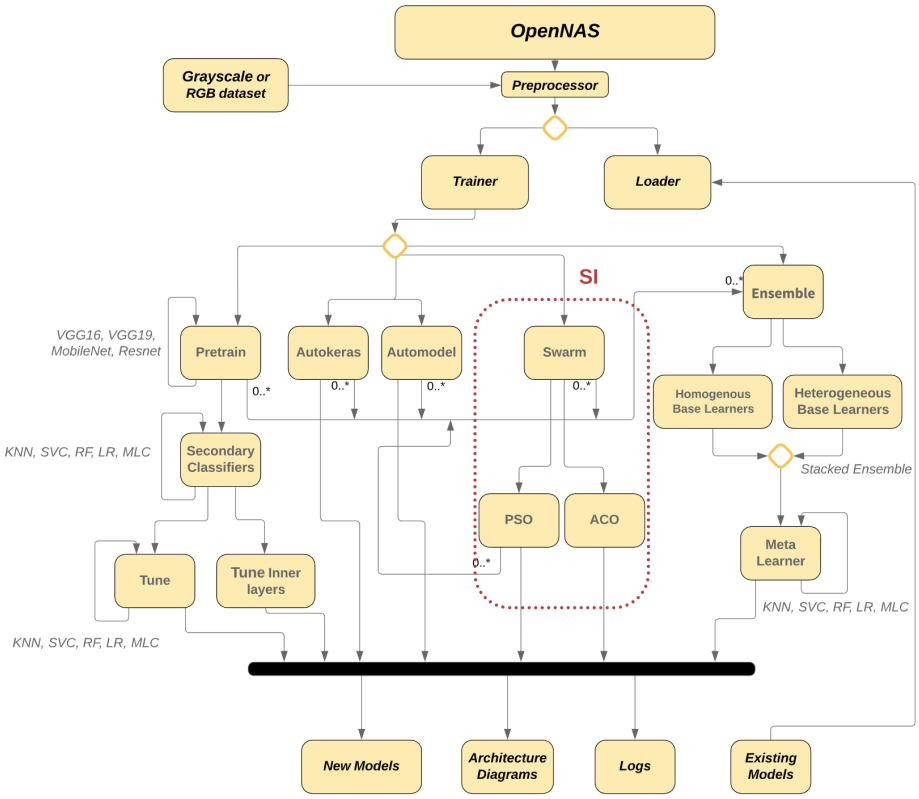

Fig. 2. Open source Neural Architecture Search (OpenNAS) System Design

4 Design

The high level view of the system architecture is presented in Figure 2. The system is organized into the following python modules: OpenNAS, pre-processor, trainer, ensemble, super stacker, sysconfig and loader. The pre-train function uses transfer learning as either a feature extractor or to fine tune the pre-trained networks of VGG16, VGG19, MobileNet or ResNet50.

With the swarm function, PSO or ACO can be used to search for the best neural architecture. Existing open source python libraries were customized for both PSO and ACO functionality. Particle swarms were implemented using a psoCNN library [18] whereas ant colonies used the DeepSwarm library [19]. The environment required Python 3.7, Tensorflow 1.14, Keras 2.2.4, Numpy 1.16.4 and Matplotplib 3.1.0.

Existing NAS tools, such as AutoKeras, were also integrated into the OpenNAS system. AutoKeras is a powerful open source library which provides functions to automatically search for optimal architectures for deep learning models.

However, this library is still in beta development and the associated documentation is quite poor.

With the ensemble module, there are options to build stacked ensembles using either homogeneous or heterogeneous base learners. These learner outputs are subsequently passed to a suite of meta learner algorithms. The system generates the optimal neural architecture model using the chosen heuristic.

5 Evaluation

Two datasets were chosen for the experimental design, namely CIFAR10 [20] and Fashion_Mnist [21]. A primary research objective is the development of a Neural Architecture Search tool which generates high performing architectures for generic datasets of either grayscale (one channel) or colour (triple channel) images. The CIFAR10 dataset meets this requirement in that it is a challenging dataset of colour images. The Fashion_Mnist dataset is also suitable since it a well-tested and well understood dataset of black and white images. For reference, the state of the art (SOA) accuracy achieved on CIFAR10 is 98.5% [22] whereas with Fashion_Mnist, the SOA accuracy is 94.6% [23].

5.1 Particle Swarm Optimization

In order to test variance and reproducibility, each configuration was run 5 times on both CIFAR10 and Fashion_Minst which resulted in the evaluation of 4000 CNN architectures for this phase of the study.

Evaluation of models trained on CIFAR10 dataset Validation accuracy was used to evaluate the performance of both PSO configurations. It is clear from Table 3 that the PSO model trained on swarm settings of a lower population and higher number of iterations (population of 10 and 20 iterations) performed significantly better. In terms of accuracy, the mean performance was 3.5% better.

Table 3. Performance of PSO models on CIFAR10

Model	Acc Max	Acc Mean	Acc StDev	Time (min)	Layers
Population 10 Iterations 20	0.900	0.853	0.044	1316	30
Population 20 Iterations 10	0.883	0.818	0.053	1119	38

Both configurations have a very low standard deviation for model accuracy indicating a high level of reproducibility between test runs. At a mean run time of 21.9 hours for the first configuration and 18.6 hours for the second configuration, the PSO search for CNN architectures is a slow process considering high

performance workstations, with NVIDIA GeForce GTX 1080 Ti graphic cards, were used.

Evaluation of models trained on Fashion_Mnist dataset PSO models trained on the Fashion_Mnist dataset (Table 4), achieved much higher accuracy compared with models developed using CIFAR10 data. Similar to CIFAR10, the low standard deviation associated with both implementations of Fashion_Mnist models indicate the PSO approach produces consistent results between different test runs.

Table 4. Performance of PSO models on Fashion_Mnist

Model	Acc Max	Acc Mean	Acc StDev	Time (min)	Layers
Population: 10 Iterations: 20	0.943	0.932	0.009	994	30
Population: 20 Iterations: 10	0.943	0.935	0.008	1319	38

The stochastic nature of metaheuristics impacts the run times associated with PSO for both Fashion_Mnist and CIFAR10. In all tests, no clear pattern emerged with regard to run times: CIFAR10 was faster using a population of 10 with 20 iterations whereas Fashion_Mnist was faster with a population of 20 with 10 iterations. Therefore, in terms of run time, no clear conclusion could be drawn by doubling the population and halving the iterations.

With regard to the impact of swarm settings on model accuracy for Fashion_Mnist, again there is little to separate the configurations. With a mean accuracy of 93.5% for a population of 20 with 10 iterations and a corresponding mean model accuracy of 93.2% using a population of 10 with 20 iterations, no clear conclusion can be drawn.

Therefore, unlike CIFAR10, changing the swarm settings by doubling population and halving iterations does not impact model accuracy in the case of Fasion_Mnist. Both configurations for the PSO algorithm perform well on this dataset.

5.2 Ant Colony Optimization

Similar to other metaheuristics, there are several parameters which can be tuned for optimal neural architecture search using Ant Colony Optimization [17]. With OpenNAS, users may select the options of depth, number of ants and number of epochs in directing how the neural architecture search is conducted.

Evaluation of models trained on CIFAR10 dataset With CIFAR10 data, the results from Table 5 indicate that a greater number of ants leads to higher

model accuracy. The improvement in max model accuracy achieved, through doubling the number of ants and halving the number of epochs, was modest at just 1.2%. The impact on run time for a small increase in accuracy was severe. Doubling the number of ants effectively doubled the run time (even though the number of epochs was halved). The standard deviation for accuracy is very low indicating good reproducibility between the various test runs.

Table 5. Performance of ACO models on CIFAR10

Model	Acc Max	Acc Mean	Acc StDev	Time (min)	Layers
Ants: 8 Epochs: 30	0.848	0.822	0.025	541	18
Ants: 16 Epochs: 15	0.836	0.821	0.014	1004	16

Evaluation of models trained on Fashion_Mnist dataset The performance of ACO models using Fashion_Mnist data is highlighted Table 6. It can be seen that both configurations perform well resulting in accuracies greater than 93%. The difference in mean model accuracy between configuration A (8 ants and 30 epochs) is trivial when compared to configuration B (8 ants and 30 epochs). However, similar to ACO on CIFAR10, the difference in run time is very significant for configuration B. Effectively it took over 7 hours longer to achieve an accuracy improvement of 0.1%.

Clearly in the case of a simpler dataset such as Fashion_Mnist, using a number of ants in excess of 8 is not worth doing. This finding is similar to that seen with the more complex CIFAR10 dataset, above. Therefore choosing the number of ants, used for this ACO implementation, is an important consideration impacting run time performance. As anticipated, the standard deviation for accuracy is also very low indicating good reproducibility between the various test runs.

Table 6. Performance of ACO models on Fashion_Mnist

Model	Acc Max	Acc Mean	Acc StDev	Time (min)	Layers
Ants: 8 Epochs: 30	0.934	0.931	0.002	375	7
Ants: 16 Epochs: 15	0.934	0.932	0.004	837	19

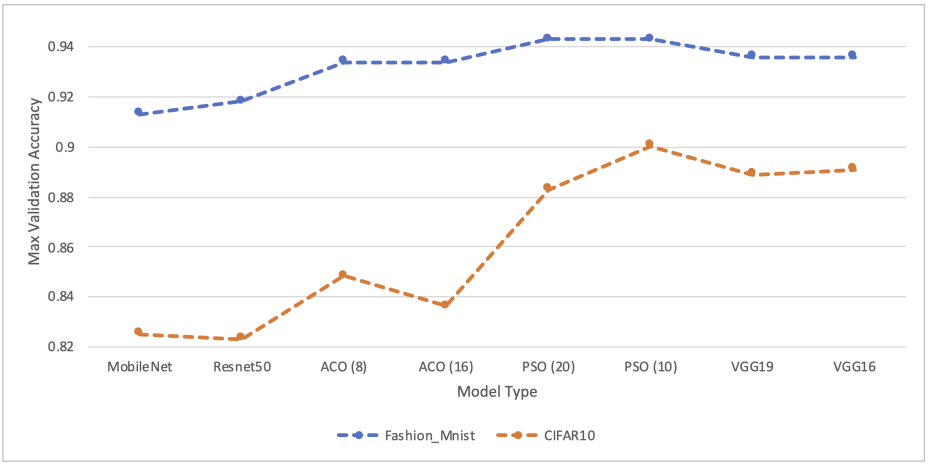

Fig. 3. OpenNAS Performance of all Models on CIFAR10 and Fashion_Mnist

6 Discussion

The OpenNAS performance of all models across both datasets is illustrated in Figure 3. The results demonstrate performance comparable to that achieved by the pso-CNN [18] approach and better than that of DeepSwarm [19].

The highest accuracy of OpenNAS in CIFAR-10 classification was 90.0%. This was achieved using using a PSO-derived model. By comparison, DeepSwarm achieved a top accuracy of 88.7%.

With Fashion_Mnist data, the highest performing model for OpenNAS is again a PSO derived model with an accuracy of 94.3%. This result compares very favourably with the SOA accuracy of 94.6%. The highest performing model for DeepSwarm achieved an accuracy of 93.56%.

In the case of pso-CNN, experiments were conducted on Fashion_Mnist but not on CIFAR-10. The best performing pso-CNN model on Fashion_Mnist was 91.9% without dropout and 94.5% with dropout.

The findings clearly show that a PSO approach leads to higher model accuracies given that DeepSwarm is exclusively based on an ACO approach.

The pre-trained networks of MobileNet and RestNet50 delivered the poorest performance with CIFAR10. The other pre-trained networks, using VGG architectures, performed very well on the same dataset.

With a more complex dataset, such as CIFAR10, the mean performance improvement of the PSO algorithm is significant when compared with ACO. With configurations used in this study, PSO achieved a mean accuracy of 85.3% on CIFAR10 compared with an ACO mean accuracy of 82.2%.

The approach taken by ACO, in determining the best architecture is very different to the PSO approach. With ACO, simpler models are initially evaluated at lower depths with progressively more complex models being evaluated at deeper search levels. Therefore at search depth 1, there is essentially just a

single hidden layer being evaluated. The number of ants specified creates new architectures which simply vary the hyper parameters used for that layer. With each new depth being explored, an additional layer is added to the architecture being explored.

Furthermore, the ACO approach enables the targeting of hyper parameter optimization within a given layer type rather than optimizing at the overall architecture level. Specifying a large number of ants, with a reduced depth, ensures the search space is restricting to studying the effects of layer hyper parameters rather than model depth and the constituent layers. By comparison, the number of layers in PSO generated models is entirely stochastic.

7 Conclusion

The OpenNAS approach identifies the hyperparameters within each layer of networks used for image classification of grayscale and color datasets. In addition the number and type of layers for the neural architecture are also identified. This combined approach generates model architectures which achieve competitve accuracies when classifying the CIFAR10 and Fashion_Mnist datasets.

The results of swarm intelligence algorithms, in the context of this study, have generated impressive performances. However, in many cases, their performance is only marginally better than fine tuned pre-trained VGG models. The accuracies of PSO derived models have been shown to exceed those of ACO derived models in the image classification of grayscale and color datasets.

In addition, the OpenNAS integrated approach, using both PSO and ACO algorithms, yields higher accuracies when compared with DeepSwarm which relies on a single metaheuristic.

References

1. F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.
2. L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 826–830, 2017.
3. B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn," in *Automated Machine Learning*. Springer, Cham, 2019, pp. 97–111.
4. H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1946–1956.
5. M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Auto-sklearn: efficient and robust automated machine learning," in *Automated Machine Learning*. Springer, Cham, 2019, pp. 113–134.
6. M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-sklearn 2.0: The next generation," *arXiv preprint arXiv:2007.04074*, 2020.
7. R. S. Olson and J. H. Moore, "Tpot: A tree-based pipeline optimization tool for automating," *Automated Machine Learning: Methods, Systems, Challenges*, p. 151, 2019.

8. B. A. Garro and R. A. Vázquez, "Designing artificial neural networks using particle swarm optimization algorithms," *Computational intelligence and neuroscience*, vol. 2015, 2015.
9. M. Mavrovouniotis and S. Yang, "Training neural networks with ant colony optimization algorithms for pattern classification," *Soft Computing*, vol. 19, no. 6, pp. 1511–1522, 2015.
10. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
11. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
12. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
13. T. Elsken, J. H. Metzen, F. Hutter *et al.*, "Neural architecture search." 2019.
14. V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.
15. J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.
16. R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *International conference on evolutionary programming*. Springer, 1998, pp. 611–616.
17. M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.
18. F. E. F. Junior and G. G. Yen, "Particle swarm optimization of deep neural networks architectures for image classification," *Swarm and Evolutionary Computation*, vol. 49, pp. 62–74, 2019.
19. E. Byla and W. Pang, "Deepswarm: Optimising convolutional neural networks using swarm intelligence," in *UK Workshop on Computational Intelligence*. Springer, 2019, pp. 119–130.
20. A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, vol. 55, 2014.
21. H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
22. E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 113–123.
23. B. Ma, X. Li, Y. Xia, and Y. Zhang, "Autonomous deep learning: A genetic dcnn designer for image classification," *Neurocomputing*, vol. 379, pp. 152–161, 2020.