Evolutionary Programming based Deep Feature and Model Selection for Visual Data Classification

Haiman Tian[1], Shu-Ching Chen[1], Mei-Ling Shyu[2]
[1]School of Computing and Information Sciences
Florida International University, Miami, FL 33199, USA
[2]Department of Electrical and Computer Engineering
University of Miami, Coral Gables, FL 33124, USA
{htian005, chens}@cs.fiu.edu, shyu@miami.edu

***Abstract*—Deep Learning (DL) has made significant changes to a large number of research areas in recent decades. For example, several astonishing Convolutional Neural Network (CNN) models have been built by researchers to fulfill image classification needs using large-scale visual datasets successfully. Transfer Learning (TL) makes use of those pre-trained models to ease the feature learning process for other target domains that contain a smaller amount of training data. Currently, there are numerous ways to utilize features generated by transfer learning. Pre-trained CNN models prepare mid-/high-level features to work for different targeting problem domains. In this paper, a DL feature and model selection framework based on evolutionary programming is proposed to solve the challenges in visual data classification. It automates the process of discovering and obtaining the most representative features generated by the pre-trained DL models for different classification tasks.**

***Keywords*-evolutionary programming; deep learning; image classification; transfer learning;**

I. Introduction

Multimedia research is a unique research area that offers vibrant and dynamic solutions across individual traditional disciplines. Multimedia data, such as image, video, audio, and text, can be found in almost every research area. Many research disciplines undergo fast developments by studying those data [1][2]. Within the multimedia community, research activities that generate broad outcomes can benefit a vast amount of research topics, including multimedia content understanding, multimodal human-computer interaction, etc. Image classification, as one of the fundamental research topics in multimedia, plays a vital role in both industry and academia and keeps improving by leveraging more advanced techniques [3]. Meanwhile, high-resolution data generated by various types of devices every second conveys more information that significantly improves the quality of the visual data and thus requires more effective and efficient approaches to deliver reliable solutions. In recent decades, Deep Learning (DL) has shown high impacts in multimedia research. For example, image classification and audio recognition keep reaching higher and higher accuracies and even break the ability of human beings. Therefore, DL approaches have been widely utilized in addressing multimedia data mining challenges such as video classification.

DL approaches transform the data in different ways for the feature extraction, feature generation, and feature learning processes altogether, which consolidates the complicated training process. They take raw data as the input to establish an accurate data representation model. However, designing and building an extraordinary DL model requires both expensive computational facilities and vast volumes of training data. It is unlikely that every researcher can build a single model from scratch for each research aim. Adopting a pre-trained model and transferring the knowledge that has been learned previously can significantly reduce the training time and complexity to maintain an adequate model performance while working on a new application domain. There are many pre-trained models that were built on either the same or different training datasets, enabling them to perform Transfer Learning (TL). Thus, researchers need to make decisions on which model should be used for a specific purpose. There is no universal model that can always produce outstanding results for every research aim. In addition, for an under-discovered dataset, which layer in the selected pre-trained model could produce the most representative and discriminated features for a specific task is not determined.

Although TL brings many possibilities to lots of research areas, little work has been done to investigate methods that can automatically determine a pre-trained model with the potential high-performance layers for feature extraction [4][5]. Among those optimization/search algorithms, Genetic Algorithm (GA) and Evolutionary Programming (EP) are two potential ones to achieve this objective [6][7]. The advantage of this family of algorithms is to perform global optimization without depending on probability distributions. In this paper, the advantage of EP over GA is explored by mainly utilizing gene mutation operations for automatic pre-trained DL models and feature selection. We anticipate that the crossover operation in GA is not necessary for the process to obtain an optimal solution while considering advance mutation operations.

The rest of this paper is organized as follows. In sec-

tion II, several optimization/search algorithms that have been introduced into DL research are discussed. Section III describes the proposed workflow in details, followed by a set of experimental results in section IV. Lastly, section V concludes this paper.

II. Related Work

Since a very early age, many advanced research areas, including DL now, incorporated the concepts of genetic science [8][9][10][11] to effectively solve non-deterministic polynomial hard problems, for example, finding the optimal number of hidden layers in a deep learning model. Several researchers have concluded that GAs are superior to other searching algorithms (such as grid search) when solving the search and optimization problems by finding a set of ideal solutions near the global optimal. The experimental results in [9] indicated that a GA-assisted approach improves the performance of a deep autoencoder. Multiple weights of one neural are stored and evolved via genetic operations, and only the higher rank weights are selected for backpropagation while the rest are disregarded. Other than that, deciding the number of layers before training a neural network is critical as the structure of the network is fixed when the training process starts. In [8], the authors used GA for both input feature selection and time series model building. A GA selects the optimal number of time lags to organize the inputs and the number of hidden layers for a Long-Short-Term-Memory (LSTM) deep network model from predefined ranges of time lags and layers.

Many research disciplines adopt GAs to explore ample search space efficiently. Considering DL, particularly, the number of possible network structures increases exponentially when the number of layers varies. Xie and Yuille proposed a genetic Convolutional Neural Network (CNN) in 2017 for automated deep network structures learning [11]. Specifically, they proposed an encoding method to define a fixed-length binary string to represent different network structures. Standard genetic operations, such as selection, crossover, and mutation, are defined to select individuals that obtain a higher recognition accuracy on a reference dataset. Most recently, Miikkulainen *et al.* proposed CoDeepNEAT, an automated method, for optimizing deep learning architectures through evolutions [10]. Their method achieves performances that are comparable to best human designs using standard benchmarks regarding object recognition and language modeling tasks. Neuroevolution methods are utilized for network construction, including topology, components designing, and hyperparameters selection. They also claimed that the CoDeepNEAT method could support automated image captioning. Overall, the evolution of deep networks is a promising approach to construct deep learning models giving that available computing power is expected to be further enhanced in the future.

Different from GA, EP heavily relies on the mutation operation to generate new genes. Mutation strategies are essential for EP to discover offspring far different from the parents [12]. Comparing to a few researchers who have successfully evolved networks with reinforcement learning and long-short-term memory, very few have attempted to optimize the deep CNN structure using the evolutionary programming approaches. These approaches aim to evolve the network structure and then subsequently optimize the hyper-parameters of the network. However, a mechanism to evolve the deep network structure under the techniques currently being practiced in the manual process is still absent. The incorporation of such techniques into the chromosome level of evolutionary computing can certainly take us to better topological deep structures. Dias *et al.* identified the gap between the evolutionary based deep neural networks and the deep neural networks, and concluded some insights for optimizing deep neural networks using the evolutionary computing techniques [6].

III. Proposed Framework

In Figure 1, the overall workflow of the proposed framework is depicted. For each dataset, the raw data is organized as the input for different pre-trained DL models (e.g., N models). The models have been trained on some large datasets, such as ImageNet. Each model is designed with different numbers of layers that can be considered as a collection of feature extractors. For example, X layers in model 1 can generate X sets of features.

Gene evolution first happens in the layer selection phase. By encoding each layer's id as a unique gene, gene evolution evaluates each layer's performance on a validation dataset and generates a ranked list of potential feature candidates. This process keeps evolving for several generations to output the final ranking for the top layers. After the top feature candidates (individual layers) are selected, the feature selection phase also utilizes the gene evolution process to determine whether multiple feature sets should be included to improve the current model's performance. This process imitates the design in some of the pre-trained DL models: Concatenation happens in the middle of the model, which selects multiple early layers' outputs to form a new layer. Every gene gets a score from the fitness calculation. The top one feature combination is the final feature set for a pre-trained model. All model performances are evaluated using the best feature combination on the validation dataset through the fitness score calculation and ranking to determine the final feature set that performs the best on the current dataset.

A. Gene Evolution

The gene evolution process shown in Algorithm 1 is repeatedly used in both the layer selection phase and the feature selection phase. Differences between these two scenarios are handled by the $GeneEncoding()$ function which

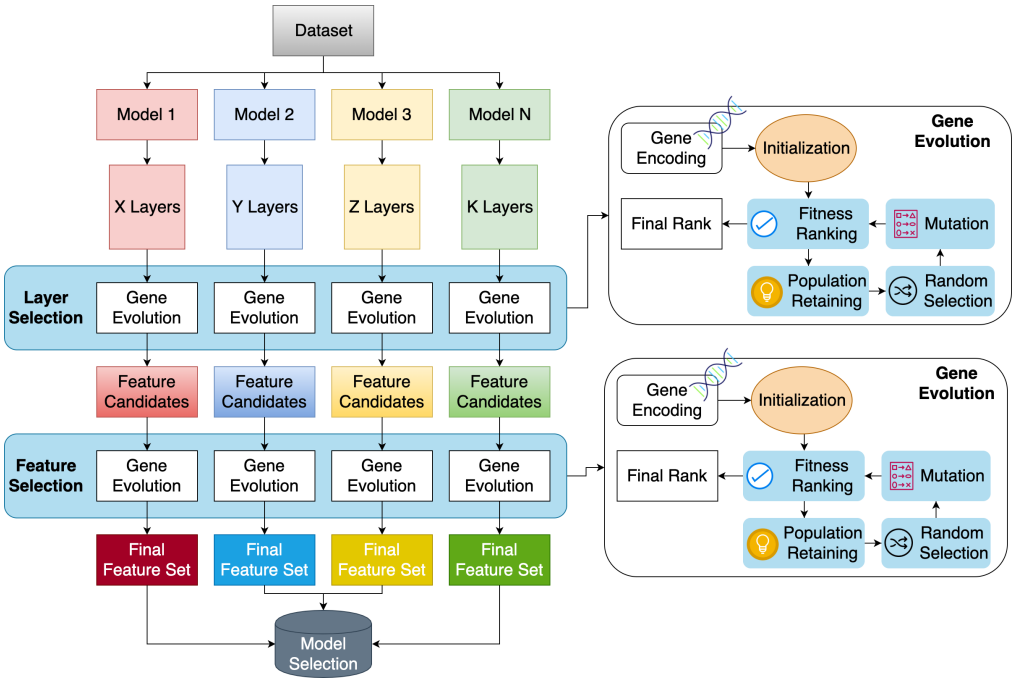

Figure 1. Proposed framework for deep learning model selection using Evolutionary Programming

sets up the restrictions for the initialization function. In layer selection, it considers the maximum number of layers that can be used for feature extraction of each specific model and converted each particular layer ID into a binary string as a unique gene to represent an individual layer. For instance, a string "11001" (binary format of decimal number 25) can represent layer ID 43 with a shift value of 18 from number 0. The length of the gene can vary for each model since the number of layers available for each model is different. Besides, the shift value is explicitly defined for each model to disregard the very first layers containing very low-level features. In the feature selection phase, one gene encodes the selection of every feature candidate as a binary value into the gene string. The length of the gene in this stage equals the total number of layers designed to be selected as potential feature extractors from layer selection. It is empirically set to be equal to the number of individuals obtained by the population retaining operation. Value 1 means a set of features is selected, while value 0 means this set of features is not chosen. Thus, string "0010" represents only the third set of features is selected for the final model training.

Three genetic operations occur in each generation (line 7 to line 23). They are population retaining, random selection, and mutation. For every generation, a fitness score is calculated for each individual in the current population. Then, the individuals are ranked in the descending order by their scores. Only the top portion of the individuals will be retained as the candidates to evolve in the next generation (40% in the proposed model). The random selection operation randomly adds some low-performance individuals to participate in the mutation operation with a 0.1 probability. The mutation operation looks through every binary value of one randomly selected gene to determine whether each piece of the gene should be changed or kept. The mutation rate is set to 50% in line 3, which means that for every binary number (a piece of gene) in a particular gene, the probability of changing or keeping the current piece is equal.

As the primary purpose of the proposed framework is to identify the best features that can be learned from the pre-trained deep learning models, linear Support Vector Machine (SVM) models are adopted for feature strength evaluation. Though more powerful classification models can replace the SVM model when considering final stage high-level feature training, the linear SVM models are efficient enough to evaluate each individual's wellness during every generation repeatedly. Hence, in the proposed framework, each SVM classifier is trained using the feature set represented by an encoded gene as the input to build a model on the target classification domain. The model's performance on the validation dataset depicts the wellness of this individual. Then, fitness score is defined as the averaged F1 score of the output among every prediction concept. This score is defined by the following equation.

$$f(i) = (\sum_{c=1}^{C} \frac{2 * P_c^i * R_c^i}{P_c^i + R_c^i})/C, \quad (1)$$

where P_c^i and R_c^i stand for precision and recall respectively

Algorithm 1: Gene Evolution

$RETAIN \leftarrow 0.4$; $SELECT \leftarrow 0.1$; $MUTATE \leftarrow 0.5$
GeneEncoding(); $Population \leftarrow$ Initialization()
for *individual* $i \in$ *Population* p **do**
    calculate FITNESS FUNCTION $f(i)$
    $grade[i].score \leftarrow f(i)$
Sort $grade$ in descending order
Population Retaining
$topGrade = grade[0 : RETAIN * grade.size]$
$restGrade = grade[RETAIN * grade.size : grade.size]$
for $x \in topGrade$ **do**
    $parents.append(x)$
# Random selection
for $x \in restGrade$ **do**
    if $SELECT > random()$ **then**
        $parents.append(x)$
$size \leftarrow Population.size - parents.size$
# Mutation
while $children.size < size$ **do**
    select $candidate$ randomly from $parents$
    for $i \in range(candidate.length)$ **do**
        if $MUTATE > random()$ **then**
            MUTATE($candidate[i]$)
    $children.append(candidate)$
$parents.append(children)$
return $parents$

Table I
THE PRE-TRAINED DL MODEL CANDIDATES WITH THE AVAILABLE NUMBER OF FEATURE CHOICES

Models	Layers	# combinations
InceptionV3	94	94^4
ResNet50	64	64^4
MobileNet	13	13^4
DenseNet201	80	80^4
Total combinations	–	$3.41E32$

for every class $c \in C$ and instance i in the validation dataset.

IV. EXPERIMENTAL RESULTS

Four pre-trained DL models currently included in the framework are listed in Table I. Each model has a different number of layers that are available for feature extraction. Since the retaining rate in the gene evolution is set to 0.4, and the number of individuals in one generation is set to 10, 4 (10*0.4) individuals are reliable for better performance in the last generation. Therefore, the feature selection phase considers combinations of the top 4 feature sets for each model. The last column in the table shows the total number of possible combinations for each model. The total number of combinations to output the final model decision reaches 3.41E32, which is not manageable by handpicking or exhaustive search.

Four visual datasets are used in the experiments to evaluate the proposed framework's performance. Two of them are balanced while the rest are imbalanced. A balance dataset shares approximately the same number of instances for every concept. On the contrary, the total number of instances for each concept in an imbalanced dataset may vary. The statistical information for the two imbalanced datasets can be found in Table II (Network Camera 10K and Disaster). Having a model to classify instances in an imbalanced dataset is more challenging since simple models are usually easy to be biased to the majority class and ignore the minority ones. In the Network Camera 10K dataset, images are captured from the surveillance camera system, and the majority concept is "Highway" which takes 1/3 of the total number of instances. The majority concepts in the Disaster dataset are "Flood and Storm" and "Speak". As the total number of instances is depicted in the table, 20% of the instances for each concept are randomly selected to form the testing dataset. Then, another 20% of the instances from the rest are further separated into the validation set. The validation data is used in the training phase to evaluate the fitness of a particular generation after training the SVM classifier. In the Disaster dataset, about 1,500 YouTube videos related to two hurricane events are used to generate the keyframe images. Each keyframe image contains one single concept to represent the entire video's semantic. In Year 2017, Hurricane "Harvey" happened first in the state of Texas and followed by Hurricane "Irma" which majorly affected Florida. The first hurricane event is used as the training dataset, and the second one is used as the testing dataset. Besides, 20% of the instances for each concept from the training set is further selected as the validation dataset. The other two balanced datasets, CIFAR10 and MNIST-Fashion, are publicly available. CIFAR 10 includes 10 classes that cover animals and objects. MNIST-Fashion is a replacement of the original MNIST dataset, which collects grayscale images of 10 clothing types.

Table III compares the proposed EP method with the other three algorithms (all from the evolutionary algorithm family). They are different in the way how specific genetic operations perform. Single position mutation EP does not incorporate crossover operations like GA. Besides that, for a length l gene code, only one position in each gene can be randomly selected for the mutation operation during each generation. GA w/o mutation excludes the mutation operations in the evolution process. Therefore, creating a new gene relies completely on the crossover operation. In regular GA algorithms, both mutation and crossover operations are included, in which the mutation operation only changes one position for one selected gene once in a generation. We report Precision, Recall, Averaged F1 score (Avg. F1), and Weighted Averaged F1 score (W. Avg. F1)

Table II
THE STATISTICAL INFORMATION OF THE NETWORK CAMERA 10K AND DISASTER DATASET

Network Camera 10K						Disaster			
No.	**Concepts**	**Instances**	**No.**	**Concepts**	**Instances**	**No.**	**Concepts**	**Harvey**	**Irma**
1	Intersection	855	**8**	Yard	161	**1**	Demonstration	42	8
2	Sky	495	**9**	Forest	139	**2**	Emergency Response	81	20
3	Water Front	978	**10**	Street	431	**3**	Flood and Storm	426	177
4	Building+Street	603	**11**	Parking	99	**4**	Human Relief	70	1
5	Park	499	**12**	Building	243	**5**	Damage	42	172
6	Montain View	719	**13**	Highway	3724	**6**	Victim	75	16
7	City	432	**14**	Park+Building	149	**7**	Speak	347	63
Total			**9527**			**Total**		**1083**	**457**

Table III
EVALUATION RESULTS ON FOUR DATASETS

Datasets	**Algorithms**	**Final Model**	**Precision**	**Recall**	**Avg. F1**	**W. Avg. F1**
Disaster	EP w/ one position mutation	InceptionV3	0.3192	0.3084	0.2514	0.3937
	GA w/o mutation	InceptionV3	0.3215	0.3256	0.2747	0.3920
	Genetic Algorithm	ResNet50	0.3212	0.3276	0.2867	0.4430
	EP Top 1 Layer	ResNet50	0.2624	0.2652	0.2302	0.3981
	Proposed EP Method	**ResNet50**	0.3186	**0.3440**	**0.2910**	**0.4432**
Network Camera 10K	EP w/ one position mutation	InceptionV3	0.6644	0.5896	0.6108	0.7705
	GA w/o mutation	ResNet50	0.6391	0.6081	0.6175	0.7761
	Genetic Algorithm	InceptionV3	0.6508	0.6339	0.6409	0.7985
	EP Top 1 Layer	InceptionV3	0.6434	0.6173	0.6278	0.7768
	Proposed EP Method	**InceptionV3**	0.6582	**0.6405**	**0.6479**	0.7957
CIFAR10	EP w/ one position mutation	ResNet50	0.8996	0.8995	0.8995	-
	GA w/o mutation	ResNet50	0.8934	0.8928	0.8930	-
	Genetic Algorithm	ResNet50	0.9063	0.9061	0.9061	-
	EP Top 1 Layer	ResNet50	0.8996	0.8995	0.8995	-
	Proposed EP Method	**ResNet50**	**0.9073**	**0.9069**	**0.9070**	-
MNIST -Fashion	EP w/ one position mutation	ResNet50	0.9282	0.9285	0.9282	-
	GA w/o mutation	ResNet50	0.9282	0.9285	0.9282	-
	Genetic Algorithm	ResNet50	0.9289	0.9292	0.9289	-
	EP Top 1 Layer	ResNet50	0.9260	0.9263	0.9260	-
	Proposed EP Method	**ResNet50**	**0.9294**	**0.9298**	**0.9294**	-

Table IV
LAYER SELECTION AND COMBINATION: PROPOSED EP V.S. GA

Datasets	**Models**		**Top 1**	**Top 2**	**Top 3**	**Top 4**	**Combine**
Disaster (ResNet50)	**EP**	**Gene**	11001	11111	01101	11011	0101
		Layer	activation_43	activation_49	activation_31	activation_45	
	GA	**Gene**	10001	10000	11111	00001	1111
		Layer	activation_35	activation_34	activation_49	activation_19	
Network Camera 10K (InceptionV3)	**EP**	**Gene**	110000	110100	001011	100011	1011
		Layer	activation_78	activation_83	activation_41	activation_65	
	GA	**Gene**	011110	101111	010111	010110	1111
		Layer	activation_60	activation_77	activation_53	activation_52	
CIFAR10 (ResNet50)	**EP**	**Gene**	11100	11111	11001	11101	1110
		Layer	activation_46	activation_49	activation_43	activation_47	
	GA	**Gene**	11111	11001	11101	11010	1111
		Layer	activation_49	activation_43	activation_47	activation_44	
MNIST-Fashion (ResNet50)	**EP**	**Gene**	10011	10000	10110	11000	0101
		Layer	activation_37	activation_34	activation_40	activation_42	
	GA	**Gene**	10011	10000	11000	10111	1011
		Layer	activation_37	activation_34	activation_42	activation_41	

for each dataset per algorithm in Table III. For each dataset, the results by only using our EP method to select the top one feature layer are also given in the row before the proposed EP method. As can be seen from the comparison results, the proposed method can select the best feature layer comparing with the first two algorithms. However, the single feature layer's strength is not comparable with the final proposed framework, which considers feature combination. Thus, the experiments demonstrate that feature combination is necessary when considering knowledge transfer across

different datasets. Both GA and the proposed EP method identify the same model for the final feature generation for every dataset. However, the EP algorithm reaches higher Recall and Avg. F1 for all datasets. Only for Network Camera 10K dataset, EP performs 0.28% worse than GA in W. Avg. F1. However, it obtains higher results for all the other three criteria. An explanation for this issue could be the follows: because the evolution process in the proposed model takes Avg. F1 instead of W. Avg. F1 for each individual's fitness calculation, W. Avg. F1 is not seeking the most optima during the training time. This can be easily obtained by redefining the fitness function.

In Table IV, we go one step deeper to analyze which feature sets are discovered when using GA and EP in our proposed framework. The top 4 layers in the final ranking list and the final feature layer combinations are compared. As can be found from the above table, for different datasets, the feature performances are notably diverse. Top layers that appeared in EP and GA models can be completely different or overlap. For example, layers id 37, 34, and 42 in ResNet50 are identified by both EP and GA in the top 4 layers when classifying images from the MNIST-Fashion dataset. Layer 42 ranks the third in the GA model but the fourth in the EP model, which depicts that one layer with better performance appeared in the EP model (layer 40). Though the final feature combination does not incorporate layer 40 in the final model, in the same numbers of generations, EP identifies a better feature combination (concatenating features from layer 34 and layer 42) that is superior to the feature combination selected in GA. Notably, after examining the feature layers selected for the final model training, sorting all layers from one model and then selecting the top layers do not always promise an outstanding performance. A combination of less powerful feature layers may outperform the top one layer's performance. Also, utilizing the top layer features, together with other features, can downgrade the model performance for some datasets. GA tends to repeat a similar portion of predecessors as the parents who gained higher fitness scores in early generations. However, EP is more capable of identifying sparse individuals from each population.

V. Conclusions

In this paper, we propose to utilize EP to perform DL feature and model selection for visual data classification. This framework considers each layer in a pre-trained DL model as an individual that carries a unique gene in the population and uses advanced genetic operations to explore the optimal solution automatically. The experimental results prove that the crossover operation is not necessary for identifying an optimal solution within a restricted number of generations. Evolving the gene for the same number of generations as GA, EP generates new genes via the mutation operation, which carries a set of more characteristic features. Regarding the final generation's individuals selected by GA and EP, EP could find better genes with higher performance while being more diverse from each other.

Acknowledgment

This research is partially supported by NSF OIA-1937019.

References

[1] L. Zheng, C. Shen, L. Tang, C. Zeng, T. Li, S. Luis, and S.-C. Chen, "Data mining meets the needs of disaster information management," *IEEE Transactions on Human-Machine Systems*, vol. 43, no. 5, pp. 451–464, 2013.

[2] S.-C. Chen, S. H. Rubin, M.-L. Shyu, and C. Zhang, "A dynamic user concept pattern learning framework for content-based image retrieval," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 36, no. 6, pp. 772–783, 2006.

[3] S.-C. Chen, M.-L. Shyu, and C. Zhang, "Innovative shot boundary detection for video indexing," in *Video Data Management and Information Retrieval*. IGI Global, 2005, pp. 217–236.

[4] H. Tian, S.-C. Chen, and M.-L. Shyu, "Genetic algorithm based deep learning model selection for visual data classification," in *IEEE 20th International Conference on Information Reuse and Integration for Data Science*. IEEE, 2019, pp. 127–134.

[5] H. Tian, S. Pouyanfar, J. Chen, S.-C. Chen, and S. S. Iyengar, "Automatic convolutional neural network selection for image classification using genetic algorithms," in *IEEE International Conference on Information Reuse and Integration*. IEEE, 2018, pp. 444–451.

[6] M. Dias, D. De Silva, and S. Fernando, "On optimizing deep convolutional neural networks by evolutionary computing," *arXiv preprint arXiv:1808.01766*, 2018.

[7] J. Liang, E. Meyerson, and R. Miikkulainen, "Evolutionary architecture search for deep multitask networks," in *Genetic and Evolutionary Computation Conference*. ACM, 2018, pp. 466–473.

[8] S. Bouktif, A. Fiaz, A. Ouni, and M. Serhani, "Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches," *Energies*, vol. 11, no. 7, p. 1636, 2018.

[9] O. E. David and I. Greental, "Genetic algorithms for evolving deep neural networks," in *Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 1451–1452.

[10] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy *et al.*, "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier, 2019, pp. 293–312.

[11] L. Xie and A. Yuille, "Genetic CNN," in *IEEE International Conference on Computer Vision*, 2017, pp. 1379–1388.

[12] C.-Y. Lee and X. Yao, "Evolutionary programming using mutations based on the lévy probability distribution," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 1–13, 2004.