Automatic Convolutional Neural Network Selection for Image Classification Using Genetic Algorithms

Haiman Tian*, Samira Pouyanfar*, Jonathan Chen†, Shu-Ching Chen* and Sitharama S. Iyengar*
*School of Computing and Information Sciences
Florida International University, Miami, Florida 33199 USA
Email: {htian005, spouy001, chens, iyengar}@cs.fiu.edu
†Miami Palmetto Senior High School, Miami, Florida 33156 USA
Email: jchen7760@gmail.com

***Abstract*—Deep neural networks such as Convolutional Neural Networks (CNNs) have achieved several significant milestones in visual data analytics. Benefited from transfer learning, many researchers use pre-trained CNN models to accelerate the training process. However, there is still uncertainty about the deep learning models, structures, and applications. For instance, the diversity of the datasets may affect the performance of each pre-trained model. Therefore, in this paper, we proposed a new approach based on genetic algorithms to select or regenerate the best pre-trained CNN models for different visual datasets. A new genetic encoding model is presented which denotes different pre-trained models in our population. During the evolutionary process, the optimal genetic code that represents the best model is selected , or new competitive individuals are generated using the genetic operations. The experimental results illustrate the effectiveness of the proposed framework which outperforms several existing approaches in visual data classification.**

***Keywords*-Genetic Algorithms; Deep Learning; Image Classification; Convolutional Neural Network;**

I. Introduction

The widespread growth of multimedia data including video, image, audio, and text has provided extensive opportunities in various big data applications [1], [2], [3], [4]. Among them, visual data analytics is a fundamental task in multimedia data. Its applications include video event detection [5], autonomous driving [6], robotics [7], healthcare [8], and disaster management [9], [10].

Nowadays, remarkable progress has been achieved in visual data analytics leveraging deep neural networks. In particular, Convolutional Neural Networks (CNNs) have been extensively used for image classification and recognition [11], [12], [13]. These accomplishments are primarily due to the powerful machines (e.g., with GPUs) and availability of large-scale annotated datasets (e.g., ImageNet). Although the existing CNN models have been shown to be effective, the networks are usually designed manually for different tasks. In addition, it is shown that different networks perform well for different tasks and datasets [5]. Therefore, it is critical to automatically decide the best network for a specific dataset.

The power of transfer learning in visual data analytics has been extensively pointed out in the literature [5], [14]. Existing deep learning models have millions of parameters which require immense computing power and very large-scale datasets to be trained from scratch [12], [13]. Transfer learning is the solution to mitigate this problem by utilizing part of the pre-trained models as the starting point of training a related task. By using transfer learning and powerful image classifiers trained on huge datasets (e.g., ImageNet), it is possible to effectively train a deep learning model on regular datasets with thousands rather than millions of training samples. However, the questions are (1) How can one determine the most efficient pre-trained model for each dataset? and (2) Will the size of the dataset or the nature of the data affect the classification results? For example, in our previous work on video event detection [5], AlexNet [11] performs better than the advanced models such as GoogleNet [12] on the disaster-related video dataset, while ResNet achieved the best performance in a public dataset called TRECVID SIN [15]. In addition, based on our preliminary results (as shown in Figure 1), Inception-v3 [16] performs well for a balanced dataset like CIFAR10 but ResNet or MobileNet [17] may perform better on imbalanced data. This inconsistency is mainly due to the level of similarity between the source (e.g., ImageNet) and target (e.g., disaster or TRECVID) datasets. Other factors such as the distribution of data, size of the dataset, and resolution of images may also affect the performance of each model. Thus, in this study, we try to answer these questions using an optimization algorithm.

Genetic algorithm (GA) is a subset of Evolutionary Algorithms (EA). It is ordinarily used for search and optimization problems using biogenetic operations such as selection, mutation, and crossover [18]. In recent years, integrating GA with deep learning has been attracting significant attention. More specifically, it is utilized for automatic selection of hyper-parameters (e.g., learning rate), parameters (e.g., kernel size), and network structures [19], [20].

Different from existing work, we utilize GA to enhance the performance of CNNs in image classification tasks using transfer learning in this work. Specifically, several existing

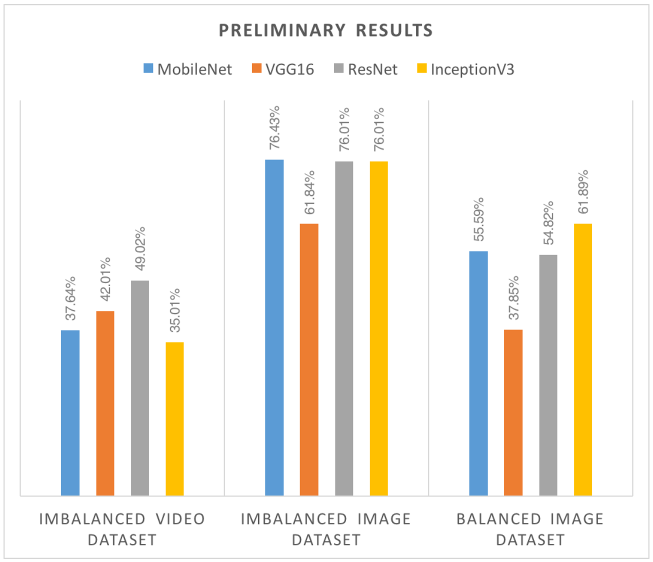

Figure 1. The accuracy of four pre-trained deep learning models on three different datasets.

pre-trained models are selected as the original population and then GA is utilized to automatically select the best model from the population or to regenerate the new candidates using GA operations. To serve this purpose, a new encoding model representing the pre-trained models in the population is presented. The GA model selects the best CNN model and extracts the corresponding features from that model. The fitness function employed in the GA algorithm is F1-score which is regularly used for the evaluation of imbalanced datasets.

To the best of our knowledge, this is the first work applying GA algorithm for automatically selecting the best pre-trained CNNs for image classification. The contribution of this work is twofold: (1) a genetic encoding model is created to improve the process of approaching optimal solutions for network selection, (2) an adaptive neural network is presented to handle both balanced and imbalanced datasets with dynamic feature inputs.

The remainder of this paper is organized as follows. In section II, the related work in the area of CNNs and GA is provided. Section III presents the details of the proposed framework. Experimental results are discussed in section IV. Finally, the paper is concluded in section V.

II. Related Work

A. Convolutional Neural Networks for Visual Data

With the emergence of deep neural networks, we have witnessed a revolution in many areas such as computer vision [11], Natural Language Processing (NLP) [21], and speech/audio processing [22]. Specifically, CNNs have shown notable improvements in visual data analytics such as image classification [23], object detection [24], and video event detection [25].

CNNs have a a hierarchical structure consisting of a cascade of linear and non-linear layers with local connectivity and weight sharing characteristics. It is originally proposed by LeCun *et al.* [26] for simple image recognition. That version of CNNs (LeNet-5) consists of two convolutional layers, each followed by a subsampling layer and finally ended with a fully connected layer for class prediction. Later, with the progress of hardware technology (e.g., GPUs), it has been widely used in many research and real-world applications [11].

In 2012, AlexNet [11] is proposed for image classification which extends the traditional CNNs and achieves the best results in ILSVRC 2012 with more than a 10% improvement in the top 5 test errors. This model utilizes the GPU implementation of CNNs together with the image augmentation and dropout techniques to handle the overfitting problem.

After that, a surge of research studies has been started to investigate the capability of CNNs in visual data analytics. Some studies mainly focus on the new structures of deep networks by introducing deeper [13], [27] and wider [12] CNNs.

Both VGGNet [27] and GoogleNet [12] were presented in ILSVRC 2014 and introduced very deep CNNs to further improve the image classification results. VGGNet, particularly, proposes a very simple model with 19 CNN layers, while GoogleNet, the winner of ILSVRC 2014, introduces a more complex module (Inception) which applies several operations such as convolution and pooling in parallel.

In 2015, ResNet [13] was proposed by Microsoft Research and achieved remarkable results in ILSVRC and COCO 2015. This model introduced residual connections in CNNs to overcome overfitting in very deep networks. Residual connections resulted in designing an ultra deep CNN with more than 100 layers.

In recent years, several extensions of successful deep learning models have been introduced (e.g, Inception-v3 [16]). Also, efficient CNNs models such as MobileNets [17] are proposed recently to be carried out on platforms with time-constraint and limited computation powers. These models and their pre-trained weights on very large-scale datasets (e.g., ImageNet) have been widely utilized in different applications. More specifically, recent studies have shown the importance of the deep features extracted from the pre-trained models using transfer learning over traditional hand-crafted features [5], [28].

B. Genetic Algorithms for Deep Learning optimization

Automatically learning the structure of neural networks has been studied for many years [29], [30]. Researchers pay significant attention to GA-based approaches to tune the network structure. Specifically, Leung *et al.* [31] proposed a method to handle both network structure and its parameters simultaneously. In that work, many network parameters were selected manually or fixed to a specific number due to the high computation costs of GA and hardware limitation. Tsai *et al.* [29], on the other hand, proposed a more robust method using the Hybrid Taguchi-Genetic Algorithm (HTGA) to enhance the traditional GA for better and faster convergence.

In recent years, by the advent of deep learning algorithms, researchers have studied the possibility of learning parameters [19], [20], network structures [32], and hyper-parameters [33] in deep neural networks using GA algorithms.

Young *et al.* [33] proposed a method called Multi- Evolutionary Neural Networks for Deep Learning (MENNDL) to optimize hyper-parameters in CNNs using GA. The fitness function used in that work is simply the testing error on the dataset after a specific number of iterations. The hyper-parameters include kernel size and number of filters in each CNN layer.

In another work, GA algorithm was used to optimize the parameters in Deep Belief Neural Networks (DBNN) for object recognition. In particular, parameters such as the number of epochs, learning rates, and hidden units in DBNN are optimized to decrease the training time and error rate of the object recognition task.

In the work proposed by Ijjina *et al.* [34], GA was used to determine the optimum weight initializations of deep neural networks. Specifically, it was applied to a CNN classifier for the task of human action recognition in order to avoid getting stuck in a local optimum solution.

In a recent work, Genetic CNN [32] was proposed to learn the structure of deep neural networks using GA automatically. To serve this purpose, a new encoding scheme was suggested which used a fixed-length binary sequence to indicate the network structure. Then, the accuracy on a reference set was used as the fitness function to determine the quality of each individual in a population.

Different from existing work, we propose a new encoding model which represents different pre-trained models in our population and apply GA to select the best model or generate new competitive individuals using genetic operations.

III. Proposed Framework

Figure 2 depicts the overall structure of the proposed framework which includes the genetic code generation and revolution process. In each generation, a set of pre-trained CNN models are selected. The items included in a particular set is determined by the individual's genetic code. Consequently, one individual represents a possible combination of CNN models that will be utilized to generate the deep features. Then, a linear SVM classifier is trained on the simply concatenated deep features to validate the effectiveness of the model selection using a fitness function. The fitness function defined in the genetic model takes the average F1-score from the validation data as the feedback to evaluate the individual's rank in the current population. After several iterations of genetic operations, the best individual with the highest fitness score will be selected as the optimal solution for our problem. Then, an automatically created network with several dense layers is employed to leverage the deep feature representations and generate the final classification results.

A step-by-step explanation of the proposed framework is described as follows.

Algorithm 1: Genetic code evolution

```
1  $RETAIN \leftarrow 0.2$
2  $RANDOM\_SELECT \leftarrow 0.1$
3  $MUTATE \leftarrow 0.2$
4  for $individual\ i \in Population\ p$ do
5      calculate Fitness function $f(i)$
6      $grade[i] \leftarrow f(i)$
7  Sort $grade$ in descending order
8  for $x \in [0, RETAIN * grade.size - 1]$ do
9      $parents.append(grade[x])$
10 # Random selection
11 for $x \in [RETAIN * grade.size, grade.size - 1]$ do
12     if $RANDOM\_SELECT > random()$ then
13         $parents.append(grade[x])$
14 # Mutation
15 for $pa \in parents$ do
16     if $MUTATE > random()$ then
17         $POS \leftarrow Randint(0, pa.size - 1)$
18         Flip code $pa$ at positon $POS$
19 # Crossover
20 $size \leftarrow Population.size - parents.size$
21 while $children.size < size$ do
22     select $famale$ and $male$ randomly from $parents$
23     if $female \neq male$ then
24         $child = (male[0, male.size/2 - 1] +$
             $female[male.size/2, male.size - 1])$
25         $children.append(child)$
26 $parents.append(children)$
27 return $parents$
```

A. Genetic Code Revolution

Our network selection problem can be compared to a black box switching problem. Suppose, the black box device contains a bank of four input switches which refer to the four selected deep learning models. The output can be represented as $F = f(s)$, where s is a particular setting of the four switches and $f(\cdot)$ is the fitness function. The objective of the problem is to set the switches to obtain the maximum possible F value. In our model, we are aiming to get the maximum Average (Avg.) F1 measure, thus, it is used as the fitness function. Considering a group of four binary numbers, a total of 16 (2^4) different combinations

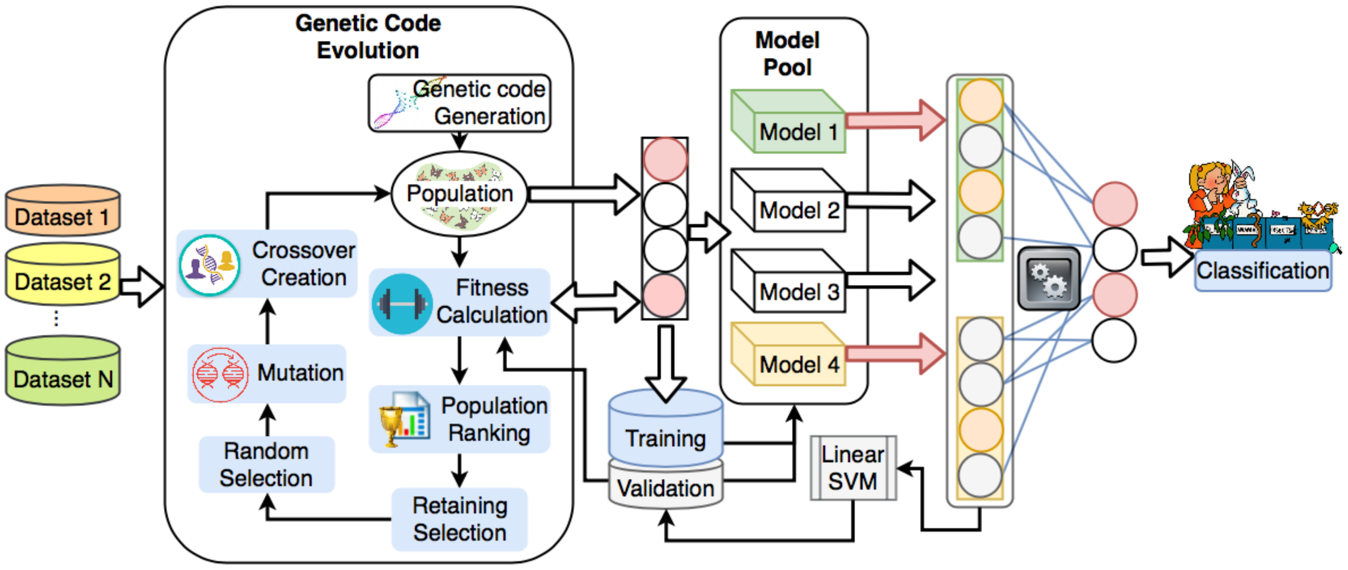

Figure 2. A genetic algorithm-based framework for deep neural network selection

from "0000" to "1111" can be generated. The combination "0000" means no model is going to be selected since there is no output signal. By removing this individual from the gene pool, there will be in total 15 different genes that lead to various output performances. Exhaustive search or brute-force search solutions are very time consuming for this problem and the complexity increases exponentially when one more model is added to the model pool. Therefore, utilizing GA is a smart method that improves the process of approaching the optimal solution in a shortcut. The entire evolution algorithm is illustrated in algorithm 1.

One iteration of the genetic code revolution consists of several genetic operations as explained below.

- **Initialization:** With GA, we first code the selected model set (switches) as a finite-length string. A simple code can be generated by considering a string of four 1's and 0's where each of the four switches is represented as "1" if the model is selected and "0" if the model is discarded. For instance, with this schema, the string "1001" encodes the setting where the first and the last switches are on while the others are off. First, the initial population is randomly selected. Then, we define a set of genetic operations that takes this initial population and generates successive populations. The new populations can be potentially improved over the time.
- **Fitness Calculation:** As we mentioned before, the function f can be considered as a measure of profit, utility, or goodness that we want to maximize. Copying genetic code according to the fitness values means a code with a higher value has a higher probability to contribute to the next generation's offspring. This operator is an artificial version of natural selection, Darwinian survival theory. For each generation, we take a portion of the best performing individuals as judged by our fitness function. These high-performers will be the parents of the next generation. Instead of running through the whole framework and getting the feedback from the validation set, we proposed to train a linear SVM classifier before building the deep neural networks as a shortcut to validate each individual in the current population. In this case, we can significantly reduce the computational complexity. Based on our preliminary results, Linear SVM classifier outperforms all the other simple classifiers, such as decision tree, RandomForest, etc. Therefore, it is utilized to calculate the fitness score and potentially ensures the reliability of the final output. Since we aim to tackle the multi-class classification task in this paper, the evaluation metrics and SVM classifier are evaluated based on the one-vs-rest decision function. The fitness function $f(s)$ for each dataset that includes C classes is calculated in Equation 1.

$$f(s) = (\sum_{c=1}^{C} \frac{2 * truePostive_c}{2 * truePositive_c + False_c})/C, \quad (1)$$

where $truePostive_c$ and $False_c$ represents the number of instances that are correctly predicted as concept c and the total number of wrongly classified instances, respectively.
- **Grades Ranking and Population Retaining:** Each individual in the current generation will get a rank based on the descending order of the fitness scores. The retaining rate is set as 20%, which means among a total number of 10 individuals in a one-time population, only the top two individuals will survive while all the

others might die. As each generation of a population is a fixed number, eight new individuals will appear according to the natural selection theory.

- **Random Selection:** We also randomly select some of the individuals with low scores as the parents, because we want to promote genetic diversity. It is very likely that optimization algorithms get stuck at a local maximum. Consequently, it may not reach the global maximum. By including some individuals who are not performing well, we decrease our likelihood of getting stuck. The random selection threshold is set to 0.1. Whenever a random number in the range of [0, 1] is generated, the number is compared with the probability threshold (e.g., 0.1). If the random number is larger than the threshold, then the corresponding individual will be temporarily kept in the gene pool. For all the genes with low fitness scores in the current population, a random selection procedure will be used to determine whether we keep the current individual or not.
- **Mutation:** Finally, mutation happens in a small random portion of the population which randomly modifies each individual. Like random selection, it also aims to encourage genetic diversity and avoids getting stuck at local maximum. As we only keep very few good individuals in each generation, we want to set the mutation probability considerably higher to speed up the revolution. For each individual in our genetic code, there are four possible positions that mutation might happen. For each individual, we restrict the operation to only change one position in one individual with a probability of 0.2. The position is also determined by a random integer ranging from [0,3]. The mutation will only flip the selected digit either from 1 to 0 or from 0 to 1.
- **Crossover:** If there are still empty slots left after retainment and random selection, a crossover will happen and fill out all the left portions. After reproduction of a new generation, the simple crossover may proceed in two steps. First step is the random pair selection. Second, each pair undergoes the cross over operation as follows: We take the first half digits from the male and the last half digits from the female. It is possible to have one parent breed multiple times, but the male and female parents cannot be identical. If the revolution process reaches a single optimal solution in an early stage, there will be no candidates remaining to process the crossover, since both female and male will be always the same. Then, the new generation will stop with fewer individuals, as we only care about the top gene at the final stage.
- **Selection of the Best Individual:** With 20% survival rate (plus an additional 10% of other individuals) and 20% mutation, the evolution always takes less than three generations (10 individuals in each generation) to reach a perfect solution. If there are more than one individual in the last generation reaching to the same top fitness score, a logic "AND" operation is applied to those codes. For example, if both codes "1101" and "1100" perform the same, we will only use the features from the first two models instead of using three models, because the redundant features increase the computational complexity without a guarantee of boosting the performance.

B. Deep Representation Learning

After the genetic code evolution, we determined a varied number of deep features. It is difficult to determine a general network works for all of the combinations. Based on our experience, the feature set that contains less than 10k features could be translated into 256-dimension high-level feature neurons. Two optimizers are our candidates, namely Adam [35] and RMSprop [36]. Adam is used as the optimizer for the balanced datasets. If it is an imbalanced dataset, RMSprop is used, and one 50% dropout layer is inserted before the last output layer (softmax layer). Dropout can significantly reduce the effect of overfitting. Batch size is automatically determined according to the size of the training samples.

IV. Experimental Analysis

A. Datasets

We selected three representative datasets from different domains to evaluate our proposed idea. First, a Youtube video dataset [25] was used that represented different disaster event-related concepts and we extracted one keyframe from each video clip. Also, two image datasets were utilized for the performance evaluation. One was collected from the network cameras located in different places [23], while the other one was a well-known public dataset called CIFAR-10 [11] that classifies objects and animals.

For the datasets that are not separated into training and testing, we randomly select 20% of samples as testing and 80% as training. Specifically, Disaster video dataset was separated into training and testing based on the time the event happens (hurricane Harvey for training and hurricane Irma for testing). CIFAR-10 data already provided the training and testing data (50K for training, 10K for testing). Form the training dataset, 20% of the samples were selected as the validation set which calculated the fitness scores in the genetic code evolution to evaluate the genetic code. Also, the same validation data were used in the last stage to evaluate the performance during feature representation learning.

The statistical information of the first two datasets were listed in Table I. Both of them are imbalanced datasets. For instance, the majority class in the Network Camera 10K is "Highway". In the Disaster dataset, the concept "flood/storm" contains most of the instances in both hurricane events. The CIFAR-10 dataset consists of ten concepts including several objects (airplane, automobile, ship, and truck) and animals (bird, cat, deer, dog, frog, and horse). It is a balanced dataset as each concept contains the same number of instances.

B. Experimental Setups

The proposed framework was compared with several successful deep learning models proposed in recent years. More specifically, we selected MobileNet, ResNet50, Inception-v3, and VGG16 which are all pre-trained on the ImageNet dataset. We used linear SVM as the classifier for all the models. In addition, we showed the performance of our genetic selection combined with a linear SVM and compared it with the whole proposed framework.

The evaluation metrics used in this work include Precision, Recall, average F1-score (Avg. F1), and weighted average F1 score (AvgW. F1) which take both imbalanced and balanced datasets into account. In particular, AvgW. F1 is calculated as the weighted sum of all F1 scores that considers the number of true instances for each class. This metric is important to show the performance of each model in a multi-class classification task.

Since Disaster dataset includes only 1K training data, the batch size of training model is set as 16, while the other two datasets are trained with batch size equals to 64. Since the proposed model only contains two dense layers at the end, we set the total number of epochs to 60 and only the best model with the lowest losses will be selected. As CIFAR-10 has larger amount of data compared to the other two datasets, the total number of epochs is set to 1200 to generate better training weights.

C. Experimental Results

Table II illustrates the performance results for all the baselines as well as our proposed genetic selection and the whole framework. As it can be inferred from this table, ResNet50 usually performs better than other deep learning models such as VGG16, and Inception-v3 in imbalanced datasets (e.g., Disaster and Network Cameras). However, Inception-v3 can significantly improve the results compared to the ResNet50 in a balanced dataset like CIFAR-10. Overall, VGG16 performs poorly in all the selected datasets. MobileNet's results are very close to the ones from ResNet-50, which shows the effectiveness of this light-version model compared to the computationally-heavy models such as Inception-v3. These results show the necessity of an automatic model to select the best model or combine the best ones in a way to maximize the final classification results.

Our proposed procedure of genetic code evolution shows the capability of identifying the best model or, in some cases, a group of models to further improve the final classification results. For the Disaster dataset, the best result is identical to ResNet50, which means combining any two or more of the models together will not improve the results. However, for the other two datasets, we can observe an improvement by combining several group of features from different pre-trained models. The AvgW. F1 improves 6% and 8% for Network Camera 10K and CIFAR-10, respectively.

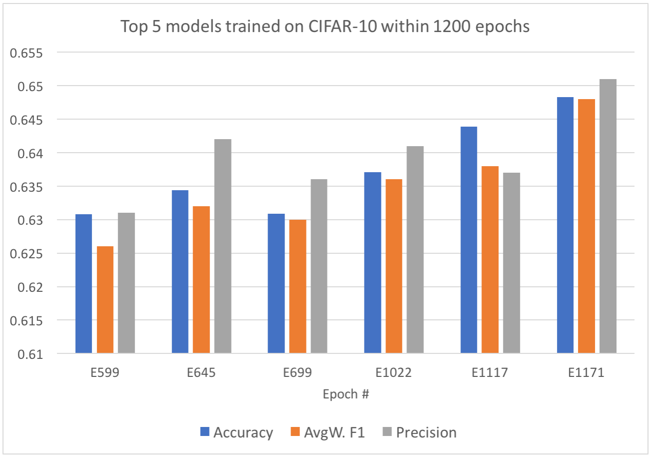

Figure 3. Top 5 models trained on CIFAR-10 within 1200 epochs

Finally, the whole framework shows more astounding improvements. We leverage the features by feeding them into another adaptive network instead of just simply concatenating them from each selected model. It can further improve the results considering all the evaluation metrics. Specifically, the AvgW. F1 reaches to 80% in the Network Camera 10K dataset. Furthermore, larger improvements can be recognized in the other two datasets.

For CIFAR-10 dataset, we also visualize the accuracy (total number of correctly classified instances), AvgW. F1, and Precision from the last five models that were automatically saved during the training process with descending losses. From Figure 3, we can conclude that, although the Precision fluctuated over the time, a trend of further improvement can be expected with more iterations.

V. Conclusion

Currently, there exist many manually designed deep learning models which are successfully applied to different tasks. However, there is no automatic way to select the best model for each dataset and domain. To address this challenge, we propose a new genetic algorithm for deep learning optimization and model selection. Specifically, the proposed genetic encoding and the adaptive network can automatically select the best model from the population. The experimental results show the effectiveness of the proposed GA method compared to other baselines.

In this work, several parameters of the adaptive network are set based on the empirical research. In the future, to enhance the capability of the network, GA can also be applied to determine all those parameters as well as the weights generated during the learning phase (e.g., the depth

Table I
THE STATISTICAL INFORMATION OF 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 II
EVALUATION RESULTS ON THREE DIFFERENT DATASETS

Datasets	Models	Precision	Recall	AvgW. F1	Avg. F1
Disaster	MobileNet	0.260	0.092	0.380	0.121
	VGG16	0.140	0.142	0.296	0.109
	ResNet50	0.296	0.113	0.419	0.141
	Inception-v3	0.197	0.071	0.303	0.092
	Genetic Selection + Linear SVM	0.296	0.113	0.419	0.141
	Proposed Framework	**0.380**	**0.136**	**0.468**	**0.163**
Network Camera 10K	MobileNet	0.610	0.145	0.755	0.216
	VGG16	0.361	0.082	0.489	0.098
	ResNet50	0.640	0.158	0.773	0.233
	Inception-v3	0.559	0.131	0.726	0.194
	Genetic Selection + Linear SVM	0.668	0.174	0.797	0.254
	Proposed Framework	**0.700**	**0.182**	**0.804**	**0.261**
CIFAR-10	MobileNet	0.446	0.083	0.446	0.14
	VGG16	0.010	0.100	0.018	0.018
	ResNet50	0.471	0.09	0.469	0.15
	Inception-v3	0.502	0.102	0.503	0.169
	Genetic Selection + Linear SVM	0.588	0.137	0.589	0.223
	Proposed Framework	**0.651**	**0.169**	**0.648**	**0.268**

and width of the network, the activation function for the dense layer, and the optimizer for the network).

ACKNOWLEDGMENT

This research is partially supported by NSF CNS-1461926.

REFERENCES

[1] C. Chen, Q. Zhu, L. Lin, and M.-L. Shyu, "Web media semantic concept retrieval via tag removal and model fusion," *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 4, pp. 61:1–61:22, 2013.

[2] T. Meng and M.-L. Shyu, "Leveraging concept association network for multimedia rare concept mining and retrieval," in *IEEE International Conference on Multimedia and Expo*, 2012, pp. 860–865.

[3] S.-C. Chen, M.-L. Shyu, and R. Kashyap, "Augmented transition network as a semantic model for video data," *International Journal of Networking and Information Systems*, vol. 3, no. 1, pp. 9–25, 2000.

[4] M.-L. Shyu, K. Sarinnapakorn, I. Kuruppu-Appuhamilage, S.-C. Chen, L. Chang, and T. Goldring, "Handling nominal features in anomaly intrusion detection problems," in *IEEE International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications*, 2005, pp. 55–62.

[5] S. Pouyanfar and S.-C. Chen, "Automatic video event detection for imbalance data using enhanced ensemble deep learning," *International Journal of Semantic Computing*, vol. 11, no. 01, pp. 85–109, 2017.

[6] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[7] H. Shahbazi, K. Jamshidi, A. H. Monadjemi, and H. Eslami, "Biologically inspired layered learning in humanoid robots," *Knowledge-Based Systems*, vol. 57, pp. 8–27, 2014.

[8] S. Liu, S. Liu, W. Cai, S. Pujol, R. Kikinis, and D. Feng, "Early diagnosis of alzheimer's disease with deep learning," in *IEEE 11th International Symposium on Biomedical Imaging*, 2014, pp. 1015–1018.

[9] M. E. P. Reyes, S. Pouyanfar, H. C. Zheng, H.-Y. Ha, and S.-C. Chen, "Multimedia data management for disaster situation awareness," in *International Symposium on Sensor Networks, Systems and Security*, 2017.

[10] H. Tian, H. C. Zheng, and S.-C. Chen, "Sequential deep learning for disaster-related video classification," in *The First IEEE International Conference on Multimedia Information Processing and Retrieval*, 2018, pp. 106–111.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[14] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[15] G. Awad, C. G. Snoek, A. F. Smeaton, and G. Quénot, "TRECVid semantic indexing of video: A 6-year retrospective," *ITE Transactions on Media Technology and Applications*, vol. 4, no. 3, pp. 187–208, 2016.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[18] C. M. Anderson-Cook, *Practical genetic algorithms*. Taylor & Francis, 2005.

[19] D. Hossain, G. Capi, and M. Jindai, "Optimizing deep learning parameters using genetic algorithm for object recognition and robot grasping," *Journal of Electronic Science and Technology*, vol. 16, no. 1, pp. 11–15, 2018.

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

[21] Y. Kim, "Convolutional neural networks for sentence classification," in *Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1746–1751.

[22] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[23] S. Pouyanfar, Y. Tao, A. Mohan, H. Tian, A. S. Kaseb, K. Gauen, R. Dailey, S. Aghajanzadeh, Y.-H. Lu, S.-C. Chen, and M.-L. Shyu, "Dynamic sampling in convolutional neural networks for imbalanced data classification," in *The First IEEE International Conference on Multimedia Information Processing and Retrieval*, 2018, pp. 112–117.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[25] H. Tian, Y. Tao, S. Pouyanfar, S.-C. Chen, and M.-L. Shyu, "Multimodal deep representation learning for video classification," *World Wide Web*, pp. 1–17, 2018.

[26] 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.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[28] M. Papakostas, E. Spyrou, T. Giannakopoulos, G. Siantikos, D. Sgouropoulos, P. Mylonas, and F. Makedon, "Deep visual attributes vs. hand-crafted audio features on multidomain speech emotion recognition," *Computation*, vol. 5, no. 2, p. 26, 2017.

[29] J.-T. Tsai, J.-H. Chou, and T.-K. Liu, "Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 69–80, 2006.

[30] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Workshop on Machine Learning in High-Performance Computing Environments*. ACM, 2015, pp. 4:1–4:5.

[31] F. H.-F. Leung, H.-K. Lam, S.-H. Ling, and P. K.-S. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," *IEEE Transactions on Neural networks*, vol. 14, no. 1, pp. 79–88, 2003.

[32] L. Xie and A. Yuille, "Genetic CNN," in *IEEE International Conference on Computer Vision*, 2017, pp. 1388–1397.

[33] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Workshop on Machine Learning in High-Performance Computing Environments*, 2015, p. 4.

[34] E. P. Ijjina and K. M. Chalavadi, "Human action recognition using genetic algorithms and convolutional neural networks," *Pattern Recognition*, vol. 59, pp. 199–212, 2016.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[36] Y. N. Dauphin, H. de Vries, J. Chung, and Y. Bengio, "Rmsprop and equilibrated adaptive learning rates for non-convex optimization," *CoRR*, vol. abs/1502.04390, 2015. [Online]. Available: http://arxiv.org/abs/1502.04390