Genetic Algorithm to use for finding optimal feature maps and kernel size for a CNN.

Mukesh Yadav, ID: 19069792, Student of Information Science at Auckland University of Technology.

***Abstract*—Convolutional neural networks (CNNs) have been used for tasks that are linked to computer vision, such as image classification, optical character recognition (OCR), object detection and so on. However, designing a CNN architecture requires a series of experiments that are performed by varying hypermeters that comes from a large search space and results in high computation and resources cost, moreover a domain knowledge expert is required to design a CNN for a particular problem.**

In this paper, we discuss how to design CNN automatically using Genetic algorithm (GA) by keeping constraints on certain hypermeters and to look for an economical solution (a smaller number of experiments) to build CNN automatically or a guiding tool to build a complex CNN. As the depth of CNN layers increases, the hyperparameters increase exponentially which makes genetic algorithm a good choice to explore a large search space. In GA, a problem of optimizing CNN model is done by encoding hyperparameters as individual and then evaluating individual's fitness with respect to optimal value while adding variation to each generation of the population using crossover and mutation process. To evaluate the suggested method, we used a CNN with constrained architecture to classify images on MNIST, Fashion-MNIST and CIFAR-10 dataset and shows an approach to find a solution which is optimal and comparable to manual architecture LeNet-5.

***Index Terms*—Genetic Algorithm (GA), Convolution neural network (CNN), Artificial neural network (ANN), Computer vision neural network optimization.**

I. Introduction

Deep Learning has been used in several domains and has exhibited an excellent performance for the tasks performed by computers which are trivial to humans and challenging for machines, such as image classification, object detection, segmentation, natural language processing (NLP), time-series data analysis, voice recognition and their variants. They all use deep neural networks that are made up of layers and in each layer, there are several perceptron wherein each perceptron is similar to individual decision machine or brain neuron and this structure is called an Artificial Neuron Network (ANN). CNN is the one of the variants of ANN which is used to perform tasks related to vision, such as image recognition, object detection in an image, segmentation and achieved state-of-the-art results. CNN is mainly composed of several convolution layers, filters(kernel), pooling layers, full connected layers and optional SoftMax layer. Last few years have witnessed for several CNN architectures that have exhibited state-of-the-art results for NLP, such as BERT [1] and GPT-3 [2]. Similarly, there are several state-of-the-art models for image classification, such as GoogleNet [3], ResNet [4] and DenseNet [5]. On the similar line, YOLO [6], SSD [7] and Faster-RNN [8] are state-of-the-art for objection detection for an image.

Since a CNN model could be composed of several layers and several hyperparameters in each layer. It becomes tedious to fine tune those hyperparameters against a specific dataset by attempting each and every permutation of all hyperparameters, therefore, a domain expert is required to fill the gap to minimize attempts and therefore, the sequence of errors and trials. However, to fill the gap of expertise needed for the construction of CNN architecture can be done by using automation design method. Implementing a CNN model can be defined as optimization problem of hyperparameters where it is required to find right number of layers, number of feature maps, kernel size, stride size, padding size, pooling layers, fully connected layers and their interconnection.

Since this is a problem of finding optimized hyperparameters in large search space and can be solved with the help of genetic algorithm. In genetic algorithm, a population is generated using individuals and follows a biological evaluation process for the progress, on upcoming generations. It involves an individual as chromosome which encodes traits as genes and performs processes, such as mutation, crossover and selection for the next generation offspring. Next, the best fitted individual can be generated by using selection, crossover and mutation by evaluating an individual's fitness value against optimal conditions for a specific dataset.

Since, training a neural network is both computational and resource expansive i.e. it requires both high-end hardware, time and money because it has to pass through a complete training process for each individual for each generation of the population and compares its fitness value to optimal value. Therefore, we decided to use light-weight data to train a constrained neural network. We used MNIST, Fashion-MNIST and CIFAR-10 datasets to train our proposed CNN architecture by using a genetic algorithm to find optimal hyperparameters. After that, we used a standard model LeNet-5 and trained it on a given dataset, MNSIT, Fashion-MNSIT and CIFAR-10 and compared their results with our proposed CNN architecture.

With this paper we will be at the position where we can use our proposed algorithm to determine optimal hyperparameters for CNN which will reduce effort of manual effort and reduce timing to get an optimal result and a guideline to generate deeper neural network. The rest of the paper is organized in such a way that, section 2 covers literature review. Section 3 covers our proposed algorithm that will be used to design GA to train our CNN architecture for finding optimal hyperparameters, Section 4 covers experimental and implementation, it covers implementation of generic algorithm, implementation of contained CNN model. It also covers our results that we obtained after experiment and presented them. In the last section 5, we presented conclusions from our experiment. Next, we added appendix for the code that we implemented and then references that we used.

II. Literature Review

Convolution Neural Networks:

Artificial Neutral Network (ANN) is a machine learning model [9] [10] [11] that is inspired from brain working. In a brain, there are large number of neurons that communicate to each other through connections among them, and to perform a particular task, some of them get activated to trigger that specific task. On the similar line, ANN has many neurons communicate with each other through weights and learn a specific task through feed-forward and back-propagation process.

Convolution Neural Network (CNN) is another variant of the ANN machine learning model. CNN works on a principal of human vision as suggested by paper [12]. In this paper, the author showed how a local receptive field in retina (visual field) could respond to certain neurons of visual cortex as shown in figure 1, some neuron's receptive fields might overlap with each other. Moreover, author showed that certain neurons react to straight lines and some other neurons for tilted lines. They also found that some neurons gather information from neighbor neurons and recognize higher/complex structure and hence, this mechanism could be able to understand complex pattern in a visual field.

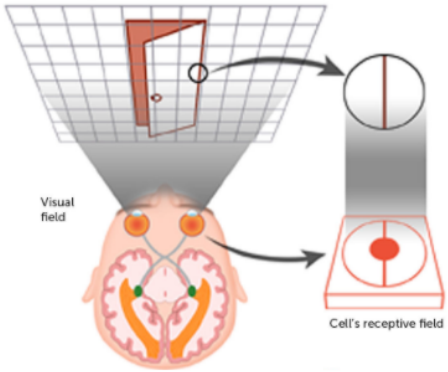

Figure 1Visual System of a Human [5]

These studies further led to neocognitron [13] which is a hierarchical, multilayered artificial neuron network and it is used for recognition of handwritten characters, this eventually led to development of a CNN architecture.

Besides fully connected layers and activation functions, CNN has convolution layers and pooling layers. Each neuron in the first layer of convolution does not connect to every input of the image like in ANN, but to all those pixels that compose the receptive field for that specific neuron in first layer of convolution. Similarly, in a second layer of convolution, a neuron is connected to small region of first layer of convolution. This hierarchical pattern helps network to learn low level features, such as vertical lines and horizontal lines, after that second layer built those low-level features into high-level features. Convolution is performed by using filters or kernel which is equivalent to receptive field of the retina. Filter is a square matrix of specified size (e.g. 3x3) with values 0 or 1 as shown in figure 2. Moving from one receptive field to another is done by using strid from left to right and from top to bottom until the whole image is parsed. A layer that is generated after parsing is called a feature map corresponding to a particular filter that was used for the convolution, and filter is not configured manually, but automatically while training the CNN to best fit to a given task, such as cat or dog classification.

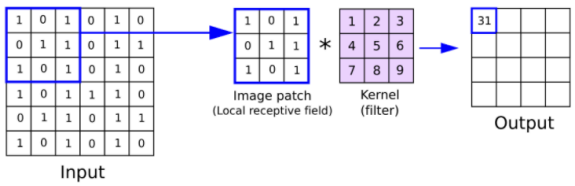

Figure 2 Convolution using 3x3 filter [14].

In the figure 2, it shows only one feature map, but in practice there could be several filters where each filter produces a feature map, therefore there would be a 3D structure. Similarly, if we consider color channel of an image i.e. RGB then filter for an instance would of size 5x5x3 where 5 is the filter size and 3 is the number of color channels. Each pixel in a feature map has one neuron with shared wights and bias. Pooling layer is used for dimension reduction, keeping dominating features and suppressing noise. Pooling is applied to a previous layer similar to the convolution using a square matrix of fixed size and stride as shown in figure 3. Max Pooling and Average Pooling are two common types of pooling, and max pooling layer is used mainly. Pooling layer has neurons but without any weights.

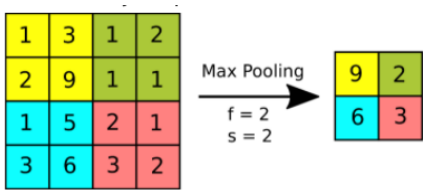

Figure 3 Max Polling Layer of size 2x2 and stride 2 [14].

After adding multiple convolution and max pooling layers, fully connected layers are added for the classification of features extracted with the help of previous layers for an image and then it is fed into SoftMax layer for the detection of probabilities of available classes. A typical CNN structure is shown in figure 4.

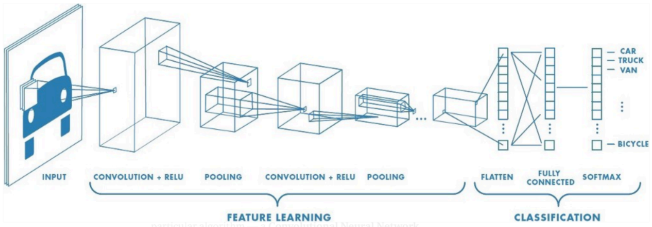

Figure 4 A typical CNN architecture [15].

Genetic Algorithm:

Evolution Algorithm (EA) is an approach to find optimal solutions for those problems that could not be solved in polynomial time i.e. NP-Hard problems. Genetic Algorithm (GA) is the subset of EA and it's based biological evolution as suggested by Darwinian theory of evolution, it includes reproduction, natural selection and survival of the fittest. GA was first published as "Adaption in Natural and Artificial Systems" [16]. Since, biological evolution theory has been worked well for the understanding of life existence through the process called survival of the fittest. It has been adapted into artificial systems and it has solved complex problems within a fixed time though the problems have large space of solutions.

In GA, first it required to encode a possible solution of the problem that needs to be solved. This encoded solution is called chromosome and it is composed of genes. A value assigned to a gene is called Allele. In figure 5, a chromosome is shown which is binary encoded. In this chromosome each gene could have

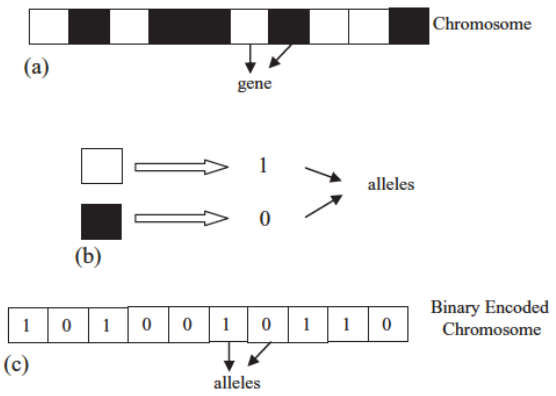

Figure 5 (a) A chromosome for a solution/individual composed of bits 1 and 0, where 1 shows presence of a trait and 0 for its absent. (b) Alleles, value assigned to a gene. (c) An array of bits will be used in GA as chromosome [17].

value 1 or 0 based on presence or absence of a particular trait. Each individual evaluated against its fitness value to achieve optimal fitness which is an objective of getting either maximum or minimum value. Most fitted individuals move to next generation as compared to lesser fitted individuals by using selection process. To produce offspring an approach called crossover is used, it generates offspring different from their parents. It is basically a process of swapping genes between two chromosomes around a point called crossover point as shown in figure 6. After this, new offspring replace two individuals from the population whose fitness values are lowest in population to keep the size of the population constant.

Mutation is another approach to create variation in a population, it does not require any of the parent. It changes the value of some of the genes with all the available values for a particular chromosome.

Selection is a process in GA to find parent members for the reproduction and it is non-deterministic process because of randomness is used for this process. A good selection process is required for the good convergence rate and for good diverse individuals for the success of GA. There are three methods that are mostly used for good results are roulette wheel selection, tournament selection and elitism.

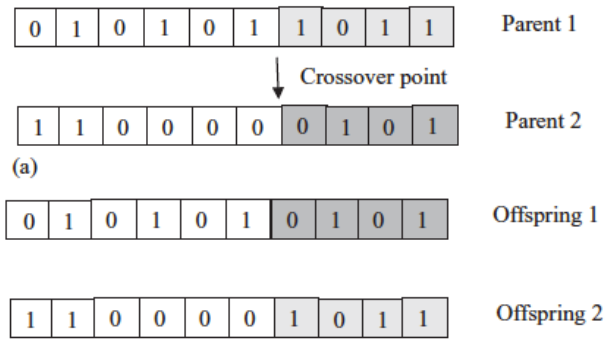

Figure 6 Single point crossover [17].

Roulette wheel selection [18] is primarily used for the selection of parents in GA for the reproduction. Roulette wheel is similar to pie-chart is divided into number of sections where number of sections is equal to number of individuals in a given population. Area of a section in a wheel depends on probability of the selection of an individual where each probability depends on the fitness of the corresponding individual. A pointer is linked

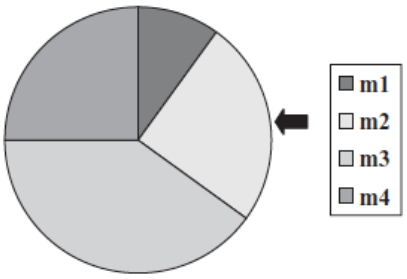

Figure 7 Roulette wheel [9]

to the wheel and whenever a wheel stops after spinning, an individual pointed by the pointer is selected for the population.

In tournament selection, some members are selected from the population and then out of those members few are selected after checking against fitness value. This process continues until the desired number of offspring are produced.

In elitism, an individual with highest fitness value moves to next generation so that it does not change by crossover and mutation, but other individuals whose fitness is lesser are selected for mating or reproduction so that they can pass to next generation after crossover.

This process of generation of offspring and selection continues until it is met to stopping criteria. Stopping criteria could be the time limit for the execution of a GA, number of generations to be produced or meeting the expectation of fitness function.

For training our proposed CNN architecture we used MNSIT dataset, it is a dataset of 60,000 images as training samples and 10,000 images as testing dataset of handwritten digits where each image is grayscale and has size 28x28 pixels. It also has labeled data for each image. Another dataset that we used is Fashion-MNSIT, it has 60,000 images as training samples and 10,000 images as testing dataset from Zalando's article images. It has a grayscale image of size 28x28 and it has data in 10 categories. CIFAR-10 dataset has colored images where each image has size 32x32x3, this dataset has size 60,000 image out of which 50,000 are used for training and 10,000 are used for testing images.

Previously [19] proposed an algorithm for designing an algorithm that would generate CNN model automatically for image classification, they devised an encoding approach where each block is composed of 2 convolutions of size of ranges 32-256, another block for pooling could be max or avg based on random condition, by this approach they have created a deeper network of linked list that is trained on CIFAR-10 dataset and compared results with other manual constructed CNN. Similarly, [20] used constrained neural network for hyperparameters optimization by encoding a desired solution as directed graph.

III. Proposed Algorithm

Algorithm 1: Proposed Algorithm

Input: A predefined CNN architecture, pre-defined set of feature maps, pre-defined set of kernel sizes, generation count, population size and dataset for which classification to be performed.

Output: Optimal parameters for the best CNN architecture specific to a task and dataset.

1. Encode an individual with desired traits (chromosome with given genes) using encoding strategy.
2. Define fitness function to calculate the fitness of an individual in a population.
3. Populate the population with number of individuals given by population size parameter.
4. Iterate steps 5, 6 and 7 from the first generation to the maximum generation that is allowed.
5. Select parents for reproduction based on fitness.
6. Perform crossover and mutation to add variation in the population.
7. Compute fitness of the population.
8. Return an array of individuals arranged in descending order of their fitness value.

Algorithm 1 is our approach to find the best CNN architecture with given constraints using GA. Since, CNN architecture can be as big and deep as it can be, therefore we added constraints to our CNN architecture to meet the scope of this paper. However, CNN architecture can be expanded to any level by following the same line of work as we have defined in Algorithm 1. We have chosen 2 layers of convolutions, first a feature map that ranges from 1 to 31 for each layer, kernel of size ranges from 1 to 7 for each layer, a dense layer (fully connected layer) and a SoftMax layer as our constrained CNN architecture.

In this Algorithm 1, we first define the strategy for the encoding of an individual. This encoding is specific to a problem and we define 16 bits array as an individual where first 5 bits represents first feature maps for first layer, next 3 bits represents kernel size for the first layer, next 5 bits represents feature maps for second layer and then last 3 bits represents kernel for second layer as shown in figure 8.

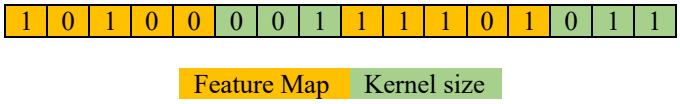

Figure 8 An individual's 16 bits representation.

To calculate fitness of an individual, we define a fitness function which takes a individual as a parameter and returns an accuracy value of the given CNN model. In the fitness function we first decode individual and extract feature maps and kernels, after that we compile our constrained CNN architecture with provided hyperparameters, and then we train our model on given training data to get fitness value that needs to be returned from fitness function.

Next, we populate the population of fixed size which is equal to parameter population size. An individual in a population is created using random combination of bits.

After that, an iterative process will start that takes initial generation and continues to follow selection, crossover, mutation and evaluation until maximum generation count is not reached. After finishing the iteration, an array of individuals sorted by their fitness value in descending order will be available.

IV. Experiments & Implementation

In order to measure performance of our proposed algorithm we used three datasets MNSIT [21] handwritten digit dataset, Fashion-MNSIT [22] Zalando's article images and CIFAR-10 [23] dataset consist of colored images. We used all three datasets with a model LeNet-5 [24] and compared their results.

For the implementation of our proposed algorithm we used a python language, a module of python called DEAP [25] for implementation of GA algorithm and TensorFlow [26] for implementation of our CNN model that need to be optimized and LeNet-5 model for the measuring of performance on given

TABLE 1
Comparisons between models obtained from our proposed algorithm and LeNet-5 model with respect accuracy of the model on training different datasets and presented best results we obtained for feature maps and kernel sizes for first and second layers.

Model/Dataset		CIFAR-10	MNSIT	Fashion-MNSIT
Our CNN Model	Feature Map (First layer)	8	31	28
	Kernel Size (First Layer)	1	5	1
	Feature Map (Second layer)	27	22	31
	Kernel Size (Second Layer)	5	2	4
	Best Individual Accuracy	0.3595	0.9031	0.7935
LeNet-5		0.2676	0.7961	0.6523

datasets to compare it with our CNN model by keeping other parameters same.

DEAP is python module for building GA algorithm, it has several functions that facilitate building of GA algorithm, one of them is the "create" function under "creator" module for creating a class, it takes at least two arguments, name for the class and the base class, we used it for the creation of class "FitnessMax" to measure the fitness of an individual and assigned a "weights" attribute to it and set its value to tuple (1.0,0) which required by base class "base.Fitness" in order for maximize fitness.

```
creator.create('FitnessMax', base.Fitness, weights=(1.0,))
```

We used the same "create" function to create another class "Individual" inherited from base class "list" and has "fitness" attribute which has value equal to class that we defined as "FitnessMax".

```
creator.create('Individual', list, fitness=creator.FitnessMax)
```

Next, we used "register" function from "toolbox" module in order to create a function called "binary", "register" function takes at least two arguments, alias and a function that is assigned to alias, here "binary" function takes "bernoulli.rvs" as alias function to be called with value 0.5 in order to generate binary data 0 or 1 by using Bernoulli Distribution, a random distribution of Bernoulli trail with probability of success is 0.5.

```
toolbox.register('binary', bernoulli.rvs, 0.5)
```

After this, we created another function called "individual" using same function "register" for an individual using class "Individual", by calling "individual" function will create a class "Individual" made up of array of bits of size specified as argument "gene_length".

```
toolbox.register('individual', tools.initRepeat,
creator.Individual, toolbox.binary, n=gene_length)
```

Similarly, we created a population of individuals using function called "population", it takes "list" as base class and generates population of size "population_size" of individuals by using function called "individual".

```
toolbox.register('population', tools.initRepeat, list,
toolbox.individual)
```

As our goal for this paper is to design the best CNN architecture for a task without the expertise level knowledge of the domain. Therefore, we followed convention over configuration for some parameters so that we, the reader of the paper can use this paper easily without delving into GA. So, we kept crossover probabilities as 40% and mutation probabilities as 10%. For selection of parents for production we used Roulette. Similarly, for our proposed CNN, we kept some parameters fixed, such as activation function set to 'relu', optimizer set to 'adam' which means learning rate set to 0.001 by default, we used 4 as epoch to run training on given dataset, loss set to 'categorical_crossentropy' and we used metrics 'accuracy' for the purpose of evaluation. Next, we fixed the number of individuals in a population by fixing parameter 'population_size' to 8 for conducting our experiments, similarly, we fixed number of generations by providing value 4 to parameter 'num_generations'. For performing experiments we used a MacBook Pro with processor core i7 2.3GHz and 16 GB 1600 MHz DDR3 RAM.

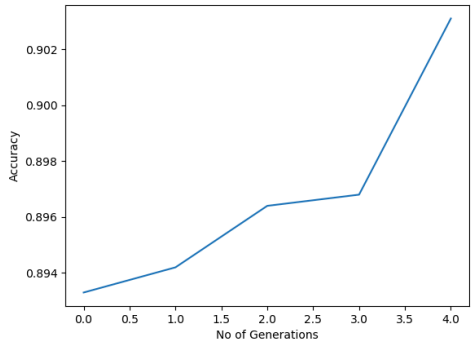

Figure 9 No of Generations vs Accuracy plot for our CNN model for MNSIT dataset.

We performed 6 experiments and presented our findings in table 1. For our first experiment, we used our proposed CNN model and MNSIT dataset, we used 10,000 images for the training for our CNN model and we presented out results in table 1 under column MNSIT. Best feature maps count we obtained for the first layer is 31 and kernel size is 5. Similarly, for the second layer we obtained 22 as best feature maps count and kernel size is 2. Our CNN model achieved 90% accuracy on MNSIT dataset. Since, this is an evolutionary algorithm that learns with every generation therefore, we recorded best accuracy value of our CNN model for each generation and

plotted as liner graph as shown in figure 9. Code for this experiment can be refer from appendix 1.

Next, we performed experiment using our CNN model and Fashion-MSNIT dataset and we kept the all other parameters same. We presented our results in table 1 under column Fashion-MNSIT. We found that for the first layer of our CNN best feature maps count is 28 and kernel size is 1 and for the second layer best feature maps count is 31 and kernel size is 4. Our CNN model achieved accuracy 79% on Fashion-MNSIT.

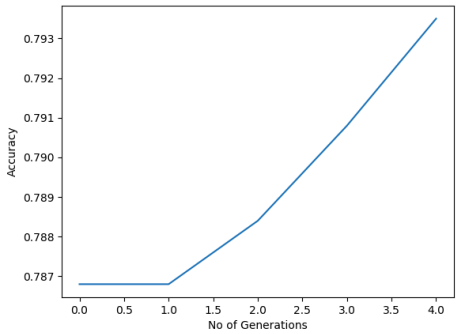

Figure 10 No of Generations vs Accuracy plot for our CNN model for Fashion-MNSIT dataset.

We plotted results of each generation's accuracy as liner graph in figure 10 as we did for MNSIT dataset. Code for this experiment can refer from appendix 2.

Similarly, we used CIFAR-10 as dataset for our third experiment with our CNN model and presented our results in table 1 under column CIFAR-10. We found that for our CNN model obtained best feature maps count and kernel size for the first layer are 8 and 1. Similarly, 27 and 5 are the best feature maps and kernel size for second layer of our CNN model by using CIFAR-10 dataset and accuracy that we obtained is 36%.

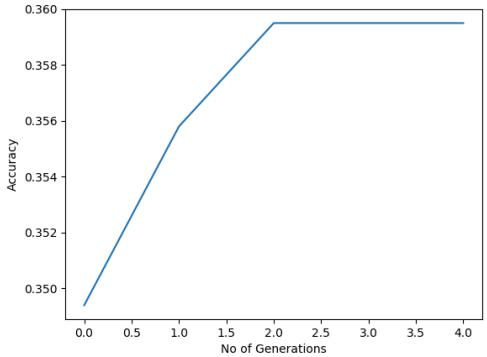

Figure 11 No of Generations vs Accuracy plot for our CNN model for CIFAR-10 dataset.

Figure 11 is a liner graph that is plotted for accuracies obtained for each generation for the experiment of our CNN model on CIFAR-10 dataset and code can be refer from appendix 3.

Our fourth experiment was performed with LeNet-5 using MNSIT dataset and presented our result in table 1 under column MNSIT and its accuracy is 79%. Its code can be referred from appendix 4. Our proposed CNN architecture with parameters determined from GA performs better than LeNet-5 model by 11%. Next, we performed our fifth experiment with LeNet-5 model and Fashion-MNSIT dataset and tabulated our result in table 1 under column Fashion-MNSIT and row LeNet-5. From this experiment, we found that this model worked with accuracy of 65% which is 14% lesser than our proposed CNN model. Code for this experiment can be referred from appendix 5. Last, we performed our experiment on CIFAR-10 dataset with LeNet-5 model and tabulated our result under column CIFAR-10 and row LeNet-5, it performed with accuracy of 26% which is 9% lesser than our proposed CNN model with parameters determined through GA and its code can be referred from appendix 6.

As shown in figure 9,10 and 11 accuracy of our proposed CNN models increase with every generation which implies that GA tends to find better solution for given CNN and dataset. Moreover, its starts with a smaller accuracy and then raises sharply with each new generation which suggests each given CNN model was initialized with random data and later on it adapts itself to better value of parameters for CNN by using GA as expected from GA. If we further look into figure 11, it suggests a saturation of accuracy after a second generation which means GA converges for this case, but it is not the case with other datasets, they require more generations for convergence.

V. Conclusion

In this paper, we proposed a GA algorithm that automatically creates CNN architecture model with constraints specific to a task, such as image classification without the expert level knowledge of the CNN domain. We took only 2 layers of convolutions and 2 filters corresponding to that layers as constraints. We encoded those constraints (layers and filters) of CNN for the optimization using GA and trained CNN models on given datasets, such as MNSIT, Fashion-MNSIT and CIFAR-10. We used LeNet-5 model for comparison to our proposed CNN model and our CNN model outperformed LeNet-5 model on given datasets for a specific task. As this evolution methodology for finding best parameters for CNN model has been resulted into saving of both time and resources, it can be used for designing deeper CNN models as an economical solution. For the future work, this approach can be evolved further by lifting constraints that we imposed so that it can handle more parameters, such as pooling layers, dropout layers, activation functions and so on. Another attribute that can be improved is uses of GPUs, parallel processing of GA algorithm and caching the results so that a bigger model and results can be constructed from smaller results that have been calculated earlier.

Appendix

1. Code for our CNN model for MNSIT dataset, https://github.com/mukeshyadav-cdac/genetic-algorithm-CNN/blob/master/mnsit.py
2. Code for Fashion-MNSIT dataset by using our CNN model, https://github.com/mukeshyadav-cdac/genetic-algorithm-CNN/blob/master/fashion_mnist.py
3. Code for CIFAR-10 dataset by using our CNN model, https://github.com/mukeshyadav-cdac/genetic-

algorithm-CNN/blob/master/cifar-10.py

4. Code for LeNet-5 model implementation using MNSIT dataset, https://github.com/mukeshyadav-cdac/genetic-algorithm-CNN/blob/master/LeNet-5-mnsit.py
5. Code for LeNet-5 model for implementation using Fashion-MNSIT dataset, https://github.com/mukeshyadav-cdac/genetic-algorithm-CNN/blob/master/LeNet-5-fashion-mnsit.py
6. Code for LetNet-5 model for implementation using CIFAR-10 dataset, https://github.com/mukeshyadav-cdac/genetic-algorithm-CNN/blob/master/LeNet-5-CIFAR-10.py

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, May 2019, Accessed: Oct. 18, 2020. [Online]. Available: http://arxiv.org/abs/1810.04805.

[2] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," *arXiv:2005.14165 [cs]*, Jul. 2020, Accessed: Oct. 18, 2020. [Online]. Available: http://arxiv.org/abs/2005.14165.

[3] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, Accessed: Oct. 18, 2020. [Online]. Available: http://arxiv.org/abs/1512.03385.

[5] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *arXiv:1608.06993 [cs]*, Jan. 2018, Accessed: Oct. 18, 2020. [Online]. Available: http://arxiv.org/abs/1608.06993.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv:1506.02640 [cs]*, May 2016, Accessed: Oct. 18, 2020. [Online]. Available: http://arxiv.org/abs/1506.02640.

[7] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," *arXiv:1512.02325 [cs]*, vol. 9905, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv:1506.01497 [cs]*, Jan. 2016, Accessed: Oct. 18, 2020. [Online]. Available: http://arxiv.org/abs/1506.01497.

[9] A. K. Jain, Jianchang Mao, and K. M. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, Mar. 1996, doi: 10.1109/2.485891.

[10] Xin Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep. 1999, doi: 10.1109/5.784219.

[11] A. Géron, "Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow," p. 851.

[12] D. H. Hubel, "Single unit activity in striate cortex of unrestrained cats," *J Physiol*, vol. 147, no. 2, pp. 226-238.2, Sep. 1959, Accessed: Oct. 15, 2020. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1357023/.

[13] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybernetics*, vol. 36, no. 4, pp. 193–202, Apr. 1980, doi: 10.1007/BF00344251.

[14] "Snapshot." Accessed: Oct. 15, 2020. [Online]. Available: https://anhreynolds.com/blogs/cnn.html.

[15] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," *Medium*, Dec. 17, 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (accessed Oct. 16, 2020).

[16] T. M. Press, "Adaptation in Natural and Artificial Systems | The MIT Press." https://mitpress.mit.edu/books/adaptation-natural-and-artificial-systems (accessed Oct. 16, 2020).

[17] A. Vasuki, *Nature-inspired optimization algorithms*. 2020.

[18] "Genetic Algorithm:A Learning Experience." http://www.cse.unsw.edu.au/~cs9417ml/GA2/selection_roulette.html (accessed Oct. 16, 2020).

[19] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020, doi: 10.1109/TCYB.2020.2983860.

[20] T. Kozek, T. Roska, and L. O. Chua, "Genetic algorithm for CNN template learning," *IEEE Trans. Circuits Syst. I*, vol. 40, no. 6, pp. 392–402, Jun. 1993, doi: 10.1109/81.238343.

[21] F. Chen, N. Chen, H. Mao, and H. Hu, "Assessing four Neural Networks on Handwritten Digit Recognition Dataset (MNIST)," *arXiv:1811.08278 [cs]*, Jul. 2019, Accessed: Oct. 17, 2020. [Online]. Available: http://arxiv.org/abs/1811.08278.

[22] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv:1708.07747 [cs, stat]*, Sep. 2017, Accessed: Oct. 17, 2020. [Online]. Available: http://arxiv.org/abs/1708.07747.

[23] "CIFAR-10 and CIFAR-100 datasets." https://www.cs.toronto.edu/~kriz/cifar.html (accessed Oct. 17, 2020).

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Ha, "Gradient-Based Learning Applied to Document Recognition," p. 46, 1998.

[25] "DEAP documentation — DEAP 1.3.1 documentation." https://deap.readthedocs.io/en/master/ (accessed Oct. 17, 2020).

[26] "TensorFlow," *TensorFlow*. https://www.tensorflow.org/ (accessed Oct. 17, 2020).