Evolutionary Painting: Use genetic algorithm and convolutional neural network to create artworks.

Weiran Liu
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, USA
weiran_liu@berkeley.edu

Huanjie Sheng
Department of Integrative Biology
University of California
Berkeley, USA
shenghuanjie@berkeley.edu

***Abstract*—In this project we investigate the ability of genetic algorithm to draw original painting. We design a selection crossover and mutation model using 50 shapes mixed with ellipse and polygon to draw MNIST-like digits given a specific target label. An MNIST classifier with 3 convolutional layers and 1 fully connected layer was used to evaluate the probably of the generated images being classified as the target label, kl....? The final results we generate are readable by human as well as incorporating nice art effect. The code of our project is available at https://github.com/vivianbuan/GeneticDraw.**

***Index Terms*—genetic algorithm, convolutional neural network, ensemble method, evolutionary painting, machine learning**

I. Introduction

Ever since the discovery of evolution by Darwin [1] in 1859, evolution has become one of the predominant theories in natural science. It provides solutions to all kinds of optimization problems in all fields of researches. Inspired by the evolutionary theory and the idea of selection, mutation and crossover, genetic algorithm (GA) has been adopted to tackle complicated tasks intangible to traditional computation algorithms [2]–[4].

Excessive studies have been done on employing GA for training neural networks [5], in which genetic algorithm is mostly considered as a searching tool in replace of or combining with gradient descent [6]. Recently, more attention has been paid on the computational creativity [7]. This leads to more and more machine learning algorithms being applied on creative domains such as music composition [8] and painting [9].

Inspired of the ability to create original art work by genetic algorithm, we would like to explore evolutionary painting using GA. Given the power of genetic algorithm, the images generated should not only be appreciated by human, but also be recognized by computers. To access the original painting generated by GA, convolutional neural networks (CNN) comes as a state-of-art tool for image classification.

Convolutional neural networks (CNN) have proved to be extremely successful for large-scale image classification [10]. The performance of current state-of-art classifier has been improved every year with a number of researchers' hard work on refining the architecture of CNN models. To benchmark the performance of models, many datasets have been made available to public, among which ImageNet [11] and MNIST [12] are two most popular one used for large-scale image recognition task and new idea testing respectively.

In this project, we start from examining the history of the interface of convolutional neural networks and genetic algorithms. Learning from the past, we design selection crossover and mutation model to draw original images using CNN image classifier as the tool to evaluate result at each generation. The effect of model dimensionality is assessed by testing with a deep and high-dimensional model, namely VGG16 built on ImageNet, as well as on a simpler model trained with MNIST. We find that given the limited computing resource we have, the system yield better result on MNIST classifier. From there, we refine the model by introducing unknown label and ensemble multiple classifiers trained with a drop-out label. As the final results, we draw MNIST-like digits of 0-9. These generated original images are readable as well as incorporating nice art effect.

II. history of cnn and ga

Genetic algorithm and convolutional neural networks share some similarities in the history of development. Both were invented in the spirit of a biological metaphor, the two algorithms were proposed ealy in the mid 19th [13]–[15] century but remained mostly theoretical until 1980s. Stepping into the 1980s, the boost in computing powers makes the two models practical. These two topics have since then remain a hit.

Neural networks are very powerful as they are universal approximators. However, the choice of hyper-parameters including model design, initial weights, and learning rates significantly influence the success of training process. The selection of those paramters is an important topic being studied by many scholars. Genetic algorithms are global search methods that are based on principles like selection, crossover and mutation [16]. Giving the natural of these two methods, the idea of combing CNN and GA came up in the late 1980s [17].

Since it first introduced, researches have investigated different combination approach: Davis [18] showed that any neural network can be rewritten as a type of genetic algorithm and vice versa, he later collaborated with Montana attempted to incorporate GA in neural network training [17]. Others

researchers have explored using genetic algorithm to perform hyper-parameter selection [19] and feature selection for neural network [20].

Although the list of studies related to combination of GA and CNN continues, most of them concentrate on using genetic algorithm as a tool to improve CNN training. In this project, we want to take the other route which takes genetic algorithm as the base and utilizes CNN to guide the evolution. Inspired by the research that uses genetic algorithm to compose music [21], and the experiment that uses genetic algorithm to draw Mona Lisa [22], we will explore the ability of genetic algorithm to draw original images given a label.

III. GENETIC DRAWING OF FRUITS AND VEGGIES

A. Flowchart of a genetic algorithm

We first construct two different models with genetic algorithms based on an existing frame work [23]. To better understand the concepts, its useful to know some biological concepts. In biology, a genotype is the genetic constitution (genome) of an organism/a cell [24]. It usually contains multiple genes, which encode the information needed to determine the characteristics of the individual. More often than not, individuals come together and form a population which will undergo selection, reproduction, and mutation. This entire process is also unknown as nature selection which is proposed by the Charles Darwin back in 1859 [1].

a) Simple Genetic Algorithm: A simple implementation of the genetic algorithm (GA) would be composed of only one genotype and one step of mutation. In our model, an individual genotype is defined as a cluster of randomly generated shapes (circles or polygons). These shapes are then sorted by the coordinates of their centroids. To speed up the simulation, one mutation is enforced after each generation which randomly reassign the location and the color of one shape (a gene) among all shapes. The new individual will survive if and only if it has a higher fitness and if the new individual survive, its parent will die immediately.

b) Selection Crossover and Mutation: On the other hand, a more complicated genetic algorithm involving selection, crossover, and mutation (SCM) as shown in **Fig. 1**. This algorithm is implemented with a sequential process of duplication, crossover, mutation and elitism. In the very first step, there is an ancestor with a randomly generated genotype and independently mutated several times to give rise to an entire population with closely related individuals. At each generation, the most fitted individuals will create imperfect replicas (with mutations) of themselves so that their good genes can be kept in the population. This process is followed by crossover between individuals. Crossover can make large scale changes than point mutation by swapping two fragments of the shape array. It can be potentially devastating if individuals are not closely related, which is also known as species isolation [25] and beneficial in terms of introducing genetic variance [26]. As is described in the previous paragraph, there is one mutation per generation to increase the rate of evolution. At the end of each generation, a few elites with the highest fitness are selected and the rest of the population is randomly sampled from the other individuals in the population. This selection process can eliminate the less fitted genotypes and keep a certain degree of variability at the same time. It should be noted that we don't have a mutation rate in our algorithm so individuals from different generations must be different.

B. Performance of two genetic algorithms

a) "Evolution" of a broccoli: To further illustrate the idea of genetic drawing, we apply the two algorithms described in the previous section III to draw a broccoli. In this demonstration, a reference image of broccoli is provided as the ground truth **Fig. 2A**. Two algorithms are randomly initiated with 250 colorful shapes on a white canvas. At each generation, the error of a genotype (a test image) is evaluated by computing the ℓ_1 norm distance of the test and the reference image, which is the opposite of fitness (Low ℓ_1 norm distance means high fitness). ℓ_1 norm is chosen to be the loss function because its more robust to outliers which are introduced during the random initiation. ℓ_2 simply squares the values so that only large pixel value difference will take priority. However, details in an image usually affect peoples view so it is more feasible to simply use the absolute difference. Additionally, a restriction on shape size is also necessary to restrict the searching space because large shapes usually dont fit the complicated features of the reference image. Polygons are only allowed to have three to five edges to reduce the computation and variance. These polygons together with circles of a similar size are drawn for equal chance at each mutation. There is no strict limitation on the number of circles or the number of polygons but the expected ratio of two shapes is one. Although polygon itself is able to make universal shape approximation, the mixture of two shapes could help depict a various of shapes in a much more efficient way [27]. Circles also have the advantage of fewer parameters limiting the degree of freedom during the evolution of test image. Finally, the reference image is resized to 80×80 to speed up the computation.

b) SCM has a lower error than SGA: With the above settings, two algorithms are run for 10000 generations. The test images (genotype) with the highest fitness (lowest error) are saved after each generation. Results at generations 0, 1000, 15000, 50000, and 100000 are presented in **Fig. 2A**. Both algorithms are able to depict the reference image fairly well from a mess of random shapes. However, SCM gets rid of the unmatched color blocks faster with only a few pieces left in the background compared to SGA. In the long run after 100000 generations, SCM does better at picking up the detail of the image. For instances, the white bottom portion of the stalk of broccoli is covered by some green color in SCM but not in SGA; the darker regions between branches have a higher contrast in SCM compared to those in SGA. The evolutionary process of error reduction is plotted in **Fig. 2B**. Both SCM and SGA have a relatively rapid descent at the first 10000 generations. SCM is able to keep a faster descending rate than SGA and reaches a lower error (ℓ_1 norm distance).

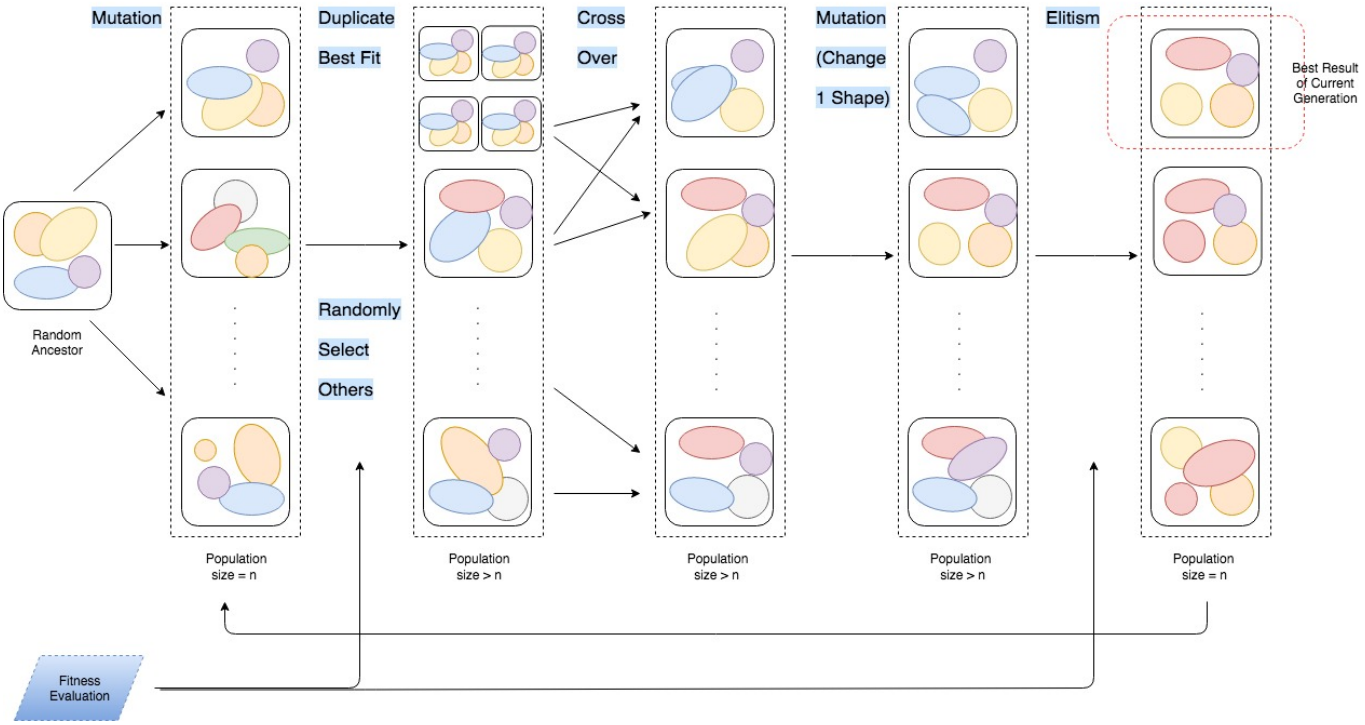

Fig. 1. Architecture of the selection, crossover, and mutation (SCM) algorithm. SCM algorithm includes five steps: random initiation, duplication, crossover, mutation, and elitism. A classification model must be chosen to evaluate individual fitness during duplication and elitism at each generation.

c) VGG16 as a evaluator: In order to make a fair comparison between SGA and SCM, we need an evaluator to score the test images generated by the two algorithms. A good evaluator must be able to accurately predict the reference image and have a broad range of class labels so the prediction is not restricted to a few classes. Therefore, we pick VGG16 in TensorFlow as our classifier [28]. VGG16 is a very deep convolutional neural network model developed by K. Simonyan and A.Zisserman to achieve large-scale image recognition [29]. It secured the first and the second places in the localization and classification tracks in ImageNet Challenge 2014 submission. We feed the best images drawn by both algorithms from all generations to VGG16 and plot the predicted probability of being classified as a broccoli in **Fig. 2C**. SCM indeed gives higher probabilities after 200000 generations with a few troughs. This might be because SCM is able to store some useful information in the individuals other than the best fitted one and retract those information later on by crossover. Alternatively, some less fitted individual could undergo a critical mutation which dramatically increase its fitness. Because there is only on individual in SGA, none of these things can happen. All these mechanisms are related to the capacity of SCM to allow for a larger searching space during evolution. Another important idea that makes SCM superb to SGA is its ability to combine genotypes and gather information. Consider a hypothetical scenario where two parts of the images are accurately described by two different genotypes. Its possible to achieve a perfect image within one step of crossover though at a relatively low probability. The large-scale change is possible because shapes are sorted by their coordinates just like genes are aligned on DNA. However, according to the no-free lunch theorem [30], the advantage of SCM comes with an enormous increase of computation complexity. Instead of blindly generating random numbers, an iteration of SCM involves the evaluations of individual fitness of the entire population. This makes SCM much slower in practice compared to SGA. Nevertheless, this increase in the computation time is still acceptable and worthy for us so we decide to choose SCM as our genetic drawing algorithm.

C. Inability of VGG16 to generate new images

We first attempt to draw a broccoli using SCM algorithm and VGG16. Specifically, the KullbackLeibler divergence (KL divergence) [31] from the probabilities of the test image to the desired one hot vector representing the image being classified as broccoli is chosen to be our fitness score. Because KL divergence cannot handle zero probability, we implement an engineering hack by setting all zero values to a very small positive number 10^{-10}. The results are shown in **Fig. 3A**. Our algorithm failed to generate a reasonable image. The 0th generation was classified as pinwheel with probability of 86.6%, the 10th generation was classified as jigsaw puzzle with probability of 5.3%, the 2000th and 5500the generation, although don't look like so, were both classified as broccoli with high probability, namely 94.1% and 99.2% respectively. A heat map of the probability matrix of the first 200 generations **Fig. 3A**

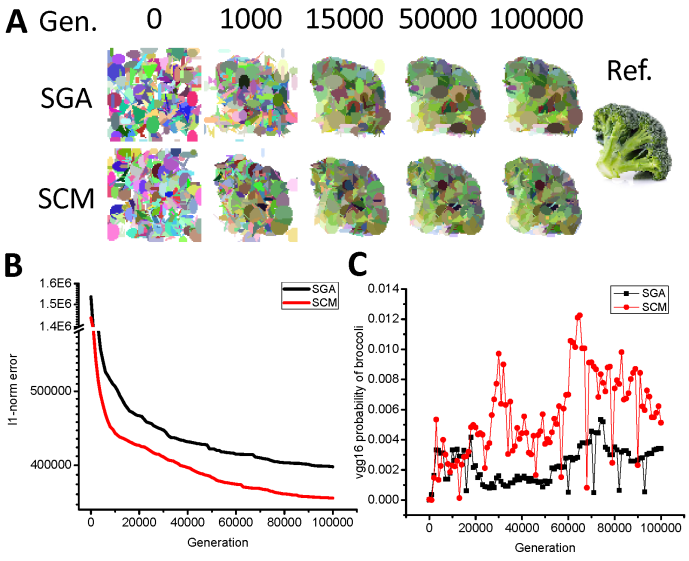

Fig. 2. Comparison between simple genetic algorithm (GA) and GA with selection, crossover, mutation and elitism (SCM) drawing with a reference image. (**A**) GA and SCM are run with a given reference image of broccoli. Generation (Gen.) 0, 1000, 2000, 5000, and 10000 are shown sequentially with the reference image. (**B**) The l1 norm distance between the reference image and the image drawn by the algorithms. Black: GA, Red: SCM. (**C**) The probabilities of being classified as a broccoli using VGG16 in TensorFlow. Black: GA, Red: SCM. It should be noted that the computation time of a generation varies between different algorithms.

An interesting observation we have from the experiment is that although the images that were drawn do not make much sense, starting from around 2000th generation, the VGG16 model classifies the generated images as broccoli with high probability ($> 95\%$). To understand this behavior, we will break down our analysis using two different approach.

a) Adversarial Image Theory: The phenomena that an image being miss-classified by CNN with high confidence reminds us of adversarial attack, which shows that changing the input slightly could significantly influence the prediction result. The VGG16 model we're using is a discriminative model which utilizes decision boundaries that partition data into classification regions. In high dimensional space, the area a model allocates to each class may be much larger than the area occupied by actual training examples, leaving plenty of rooms for adversarial images [32], or unseen images like the one we generated.

b) Neuron Concentration Analysis: To better examine why the images are classified as broccoli, we use Grad-CAM [33] to visualize the concentration of neuron during classification. As shown in **Fig. 3C**, three visualization methods–guided backpropagation (extracts edges of shapes), Grad-CAM (generates heat-map according to which feature positively contributing to the image being classified as certain label), and guided Grad-CAM (combination of the previous two methods)–were used on images generated at 0th, 100th, 2000th, and 5500th generation. We see that for generation 0 and 100, the CNN only uses corners of the image for prediction. The structure of Grad-CAM visualization for 2000th and 5000th generations looks similar: both has a stem shape at the bottom and cloud-shaped branches on top.

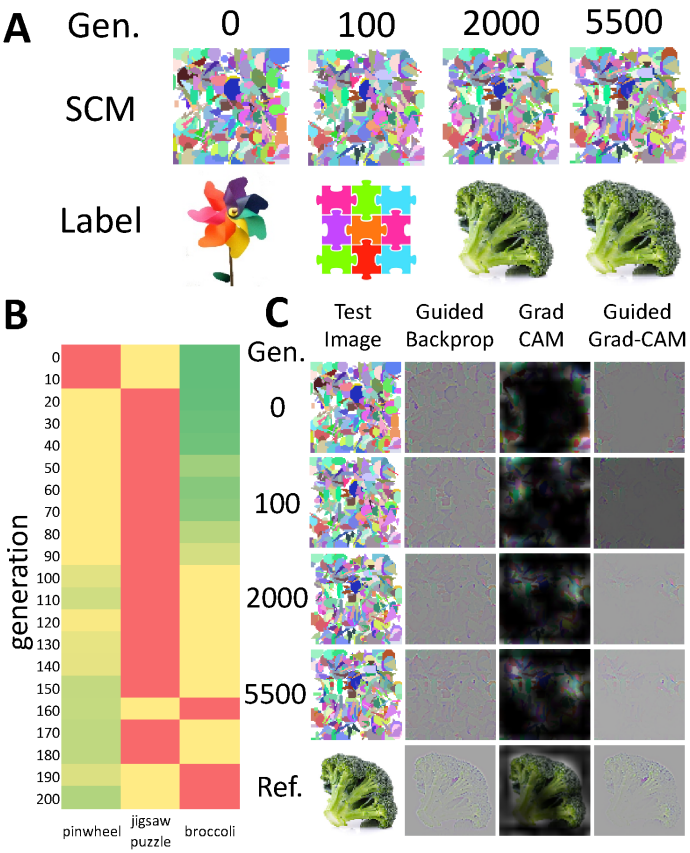

Fig. 3. Drawing with VGG16 fails due to bad local minima. (**A**) VGG16 cannot direct SCM to draw images. upper: pictures drawn by SCM with VGG16 as the predictor; lower: example images of the predicted class; from left to right: pinwheel, jigsaw puzzle, and broccoli. (**B**)The matrix of probabilities of the first 200 generations. Each row is color scaled individually by its minimum and maximum values of all classes. (**C**) Visualization of the neuron concentration to localize class-discriminative regions. Guided backpropagation: extracts edges of shapes, Grad-CAM: heat-map of which feature positively contributing to the image being classified as certain label, Grad-CAM: combination of the previous two methods.

IV. MNIST CNN MODEL TO DRAW DIGITS

A. Reduce Dimensionality compared to VGG16

Having analyzed the results of generating new images with VGG16, we identify that the main reason of the failure is that the dimensionality of the model is to high, making it easy to fall into local minima, harder to converge, and often get the model confused. To address the problem, we decided to experiment with a simpler model. Here we used an off-the-shelf MNIST classifier written by golbin [34], which is constituted of 3 convolutional layers followed by 1 fully connected layer. To further reduce the dimensionality, we changed the SCM to draw only grey-scale shapes instead of colored shapes. The input dimension of MNIST model is $28 \times 28 \times 1$ compared to the dimension of VGG16 model for which is $224 \times 224 \times 3$.

Fig. 4A shows result using 250 shapes to draw MNIST-like digits. The result has a similar trend to the previous experiment with VGG16: the images drawn start with a low probability of being identified as a digit, however, with each generation, the probably of being classified as the target digit increase rapidly, as a result, after tens of generations, images that still could not be identified by eyes are classified as 5 with high probability by the MNIST classifier. This, as we discussed earlier, was because the generated image falls into local minia.

We further explored the effect of different numbers of shapes used. As shown in **Fig. 4B**, reduce the number of shapes to 50 will significantly improve the result. This phenomena could be explained using bias-variance tradeoff. The more shapes we have, the degree of freedom increases, we have more power to explain the detail of graphs as the probability of hitting the 'desired area' increase. However, increasing number of shapes will also cause higher likelihood of shapes hitting on area that is supposed to be empty. Therefore a proper number of shapes should be chosen to ensure that we have enough explanation power but not too many bad features.

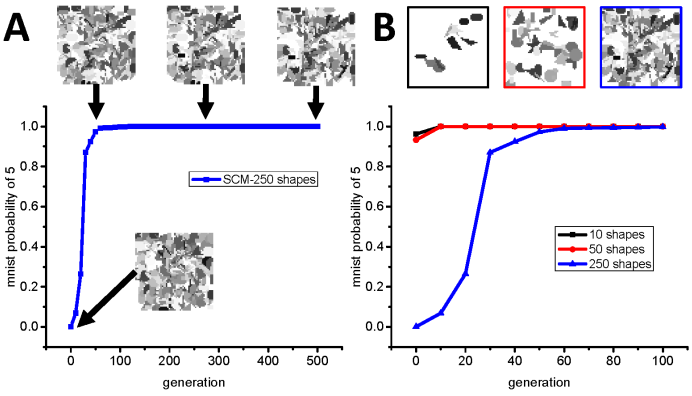

Fig. 4. Combine the genetic algorithm with MNIST CNN to draw digits. A pre-trained CNN with MNIST dataset is used to score the automatically generated images by SCM version of the genetic algorithm. (**A**) The probability of being classified as digit five quickly saturates in a few hundred generations. Four sample images are shown at generation 0, 70, 280, and 500. The number of shapes is set to 250. (**B**) The effect of the number of shapes shows a tradeoff. Only the first 100 generations are shown. Sample images of generation 500 are color framed in the same way as on the graph. Black: 10 shapes; Red: 50 shapes; Blue: 250 shapes.

B. Improving the model by unknown labels and an ensemble method

Switching from VGG16 to MNIST CNN model does generate some meaningful but still very abstract images. We then think about how to improve our classifier to make it more precise on providing feedback to the genetic algorithm. One of the problem we encountered during the trial run is that our MNIST classifier stops early and gives a very high probability to a test image with a skeleton of five but lots of random shapes as well. The most intuitive approach would be to add a new output label unknown. Adding an output gives the MNIST classifier a chance to choose an option other than the ten digits. The training process, however, is not very straightforward. The most intuitive solution is to train the MNIST classifier with some images with random shapes which are labeled as unknown. Alternative approaches like open set recognition is also helpful to classify data with unknown labels [35]. To make things simpler, we adopt the naive method and train the original MNIST classifier with images of random shapes generated in the same manner as in the random initialization of SGA and SCM. We call this model Known unknown (KU) classifier. This tweak will push GA out of the initial randomness to the right direction because most parameter space surrounding the initial test image is marked as unknown rather than a random digit. Having considered the initial direction of the evolution, we now focus on solving the issue at the end of the simulation. If a test image is good enough to be classified as a digit, then it will be eventually marked as that digit with a probability close to one. The only problem is the local minima on the path to the end. To eliminate those local minima and to pave a smoother path to the final result, we decide to use an ensemble model like random forest [36]. Random forest was first introduced to enhance the prediction power of decision trees. For our purpose, however, it is used for introducing uncertainty. In addition to the unknown label, we sequentially drop the output node of a digit and its corresponding training data to train a set of different models. One is the original model without the unknown label, and another ten are models trained with the unknown images but not one of the digit. Combining these 11 models with the KU model together gives us 12 different models. The predicted probabilities are then averaged over all models to come up with probabilities of different labels collectively. This ensemble approach gives a smoother estimation of the posterior probability which helps GA choose the path to evolve [37].

a) Knowing the unknowns: We first test the average prediction power of all three models on images drawn by SCM with a reference image **Fig. 5A**. Three models give similar trends in terms of predicting the test images of ten replicates. This is not very surprising because all three models achieves very high accuracy on either training or validation dataset. Nevertheless, if we zoom in and take a closer look, we can see that the predicted probability of the original model increases much faster than the other two models. To further illustrate this point, we repeat the same test on the images with 50 shapes from **Fig. 4B**. **Fig. 5B** clearly shows the early saturation of the original model. On the contrary, KU and ensemble models both have a slower increase at the beginning thanks to the ability to classify initial images as unknown. Its worth noticing that the ensemble model can never reach a probability of one because one of the submodel doesnt have the output of the corrected label and can never vote for it. This makes the final goal of the optimization problem unreachable. Although it seems blizzard to have an inaccessible optimal point, it actually helps GA to push the limit.

b) Disagreement in the ensemble classifier: To tackle the problem of unexpected high probabilities during latter stage in evolving the test image, we come up with the idea of using

an ensemble method. Generally speaking, ensemble methods try to average over a bunch of less powerful submodels to give a relatively more powerful ensemble model. In our case, however, we introduce the ensemble method to create uncertainty and disagreement. To make this point, we run all three models on an intermediate test image generated during the process of evolutionary painting using the original MNIST CNN model. **Fig. 5C** shows that the original and KU models give a probability of 100.0% and 99.7% respectively on classifying the image as the digit five. In contrast, the ensemble model only returns a probability of 60.8% which seems to be a more reasonable assessment of this image. The reason behind it can be readily seen from the heat map in **Fig. 5D**. Most submodels agree that the test image is indeed the digit five, while a few models that was not trained with digit five, six, seven, and eight (drop X means the model was not trained with the digit X) think this is actually an unknown image. In most classification problem, this kind of discrepancy is negative because it lowers our confidence on the predicted class label. For our purpose, however, it helps GA to obtain a collection of opinions on what the image is about rather than an arbitrary judgment by one classifier [38]. In terms of the landscape of the optimization problem, the ensemble method make it smoother removing some potential bad local minima [39].

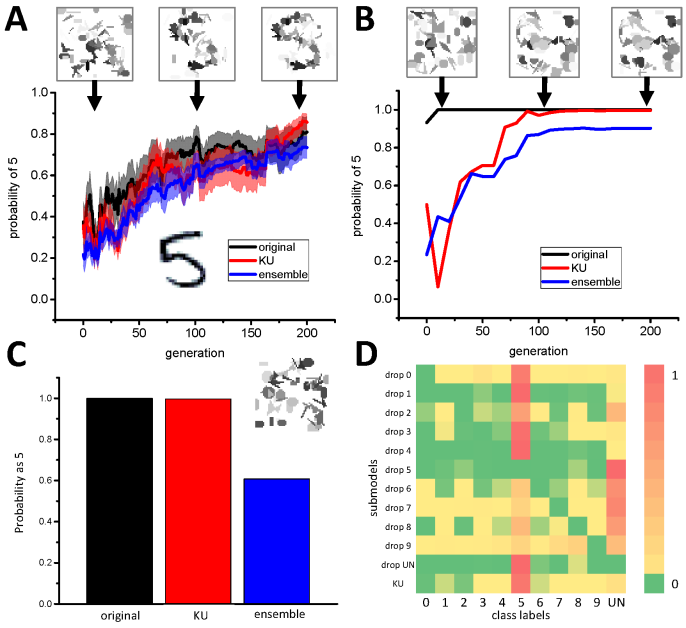

Fig. 5. Comparisons of three MNIST CNN models. (**A**) Three models perform similarly if images are drawn with a reference. Sample images are shown at generation 0, 100, and 200. The reference image is embedded in the graph. (**B**) Models trained with unknown images show lower initial probabilities. Previously generated images using the original MNIST CNN models and SCM are tested by the three models. Sample images are shown at generation 0, 100, and 200. (**C**) Random forest introduces more uncertainty due to the disagreement between submodels. (**D**) The matrix of predicted probabilities of the submodels in random forest. UN: unknown images. Black: the original model, Red: the model trained with random images labeled as unknown, Blue: A set of MNIST CNN models which might or might not drop one digit or the unknown image during the training process.

V. RESULTS

Learning from the previous experiments, we decide to choose SCM as the genetic algorithm and the ensemble model as our classifier to create our artwork. The final results of the MNIST-like digits our system generated are presented in **Fig. 6**. Compared to our previous results, the digits in final images have higher contrast to background, there are less random shapes in the background, and the overall structure of the digits are more vigorous. Although the drawings are not perfect, one can read the digits from the image. It is also important to reaffirm that we are not only trying to draw some digits, but also trying to explore a new way to create artwork. This kind of black and white painting is the essence in traditional Chinese painting, which was very popular in ancient China. We are not trying to reconstruct perfect images like those are done with generative adversarial network (GAN) [40] whose idea is also derived from biology [41], but to expand the idea of evolutionary painting associated with the knowledge in machine learning.

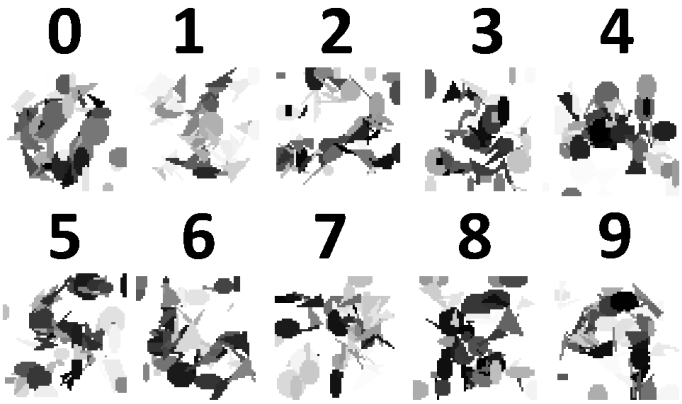

Fig. 6. Final drawing of MNIST digits using ensemble SCM model. The skeleton can be seen clearly with some art effect.

VI. CONCLUSION

Here we explore the idea of evolutionary painting which integrates genetic algorithms with machine learning models to create new artwork. We show that SCM is able to duplicate the reference image with greater details and smaller errors than SGA. In addition, we modified a MNIST CNN model by introducing an unknown label and the ensemble method to make it a better classifier to do fitness evaluation during the process of evolutionary painting. Putting together SCM and the ensemble model, we are able to create original artworks of MNIST digit in a harmony of evolution and machine learning.

ACKNOWLEDGMENT

We thank Prof.Anant Sahai for his helpful discussion on the model and Dr. Han Lim for supporting this project. Authors do not obtain special funding support for this project.

REFERENCES

[1] C. Darwon, *The origin of species by means of natural selection*. Modern Lib., 1859.

[2] E. S. Hou, N. Ansari, and H. Ren, "A genetic algorithm for multiprocessor scheduling," *IEEE Transactions on Parallel and Distributed systems*, vol. 5, no. 2, pp. 113–120, 1994.

[3] R. P. Dick and N. K. Jha, "Mogac: a multiobjective genetic algorithm for hardware-software cosynthesis of distributed embedded systems," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 17, no. 10, pp. 920–935, 1998.

[4] C. A. Coello, A. D. Christiansen, and A. H. Aguirre, "Using genetic algorithms to design combinational logic circuits," *Intelligent Engineering through Artificial Neural Networks*, vol. 6, no. 0, pp. 391–396, 1996.

[5] J. D. Schaffer, D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on*. IEEE, 1992, pp. 1–37.

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

[7] S. Colton, G. A. Wiggins *et al.*, "Computational creativity: The final frontier?" in *ECAI*, vol. 12, 2012, pp. 21–26.

[8] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," *arXiv preprint arXiv:1604.08723*, 2016.

[9] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks, june 2015," *URL http://googleresearch. blogspot. com/2015/06/inceptionism-going-deeper-into-neural. html*, 2016.

[10] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1. Barcelona, Spain, 2011, p. 1237.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[12] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[13] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[14] C. Machinery, "Computing machinery and intelligence-am turing," *Mind*, vol. 59, no. 236, p. 433, 1950.

[15] N. A. Barricelli, "Symbiogenetic evolution processes realized by artificial methods," *Methodos*, vol. 9, no. 35-36, pp. 143–182, 1957.

[16] P. Koehn, "Combining genetic algorithms and neural networks: The encoding problem," 1994.

[17] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms." in *IJCAI*, vol. 89, 1989, pp. 762–767.

[18] L. Davis, "Mapping classifier systems into neural networks," in *Advances in neural information processing systems*, 1989, pp. 49–56.

[19] D. F. Cook, C. T. Ragsdale, and R. Major, "Combining a neural network with a genetic algorithm for process parameter optimization," *Engineering applications of artificial intelligence*, vol. 13, no. 4, pp. 391–396, 2000.

[20] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," in *Feature extraction, construction and selection*. Springer, 1998, pp. 117–136.

[21] A. Horner and D. E. Goldberg, "Genetic algorithms and computer-assisted music composition," *Urbana*, vol. 51, no. 61801, pp. 437–441, 1991.

[22] R. Johansson, "Genetic programming: Evolution of mona lisa," 2008. [Online]. Available: https://rogerjohansson.blog/2008/12/07/genetic-programming-evolution-of-mona-lisa/

[23] vishaldotkhanna, "polygonga," https://github.com/vishaldotkhanna/polygonGA, 2017.

[24] W. L. Johannsen, *Om arvelighed i samfund og i rene linier*, 1903.

[25] E. E. Hunt Jr, "Animal species and evolution," 1964.

[26] S. Wright, "The genetical theory of natural selection," *Essential Readings in Evolutionary Biology*, p. 73, 2014.

[27] D. Zhang and G. Lu, "Review of shape representation and description techniques," *Pattern recognition*, vol. 37, no. 1, pp. 1–19, 2004.

[28] D. Frossard. (2016) Vgg in tensorflow. [Online]. Available: http://www.cs.toronto.edu/~frossard/post/vgg16/

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[30] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[31] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.

[32] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 427–436.

[33] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *See https://arxiv. org/abs/1610.02391 v3*, vol. 7, no. 8, 2016.

[34] golbin, "Tensorflow-mnist," https://github.com/golbin/TensorFlow-MNIST, 2017.

[35] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2317–2324, 2014.

[36] T. K. Ho, "Random decision forests," in *Document analysis and recognition, 1995., proceedings of the third international conference on*, vol. 1. IEEE, 1995, pp. 278–282.

[37] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*. Springer, 2000, pp. 1–15.

[38] P. L. Martelli, P. Fariselli, and R. Casadio, "An ensemble machine learning approach for the prediction of all-alpha membrane proteins," *Bioinformatics*, vol. 19, no. suppl_1, pp. i205–i211, 2003.

[39] A. Laio and M. Parrinello, "Escaping free-energy minima," *Proceedings of the National Academy of Sciences*, vol. 99, no. 20, pp. 12 562–12 566, 2002.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[41] W. Li, M. Gauci, and R. Groß, "A coevolutionary approach to learn animal behavior through controlled interaction," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 223–230.