Genetic Evolution of Aesthetically-Pleasing Fractals Using Convolutional Neural Networks

Kevin Yeh
CS 395T: Robot Learning from Demonstration and Interaction
Fall 2015 | Austin, TX
kevinyeah@utexas.edu

Abstract— **Computational evolution is a powerful and active field of research that allows for the automation of control tasks. However, much focus has been on the automation of tasks whose performances can be quantitatively evaluated using a fitness function. There is a need to develop and evolve methods of evaluation that can measure qualitative value, making the automation of more subjective and humanistic tasks possible. This paper proposes a human-guided approach to forming a generative model that will result in the consistent generation of diverse, aesthetically-pleasing fractals. To do so, a convolutional neural network is tuned on human-guided training feedback through an interactive genetic evolutionary program.**

I. INTRODUCTION

Fractals are defined as a curve or geometric figure, each part of which has the same statistical character as the whole"[1]. They have been researched extensively, due in large part to the complex results that can arise from iterating over simple processes many times, and fractals have been found to model a wide range of natural phenomena. Each fractal family is defined by a basic mathematical structure, from which variations are created by altering the coefficients of the basic equation. Because these coefficients can be represented by floating point numbers, determining a genome for the fractals is potentially simple and can lead to computationally light mutations, through slight alteration of the coefficients, and crossovers, through an averaging of the parents coefficients. It is also possible to expand upon the process of generating offspring fractals by taking outside factors into account, and entire families can be generated programmatically through evolution of the equations themselves, creating potentially complex fractal patterns tended towards certain desirable features.

While fractals have been researched for decades, many problems remain unsolved. One major unsolved problem is understanding what makes a fractal that is aesthetically pleasing to many human beings. Although it is difficult to make something and judge whether it will be generally well-liked among a large, diverse populus, recent advances in convolutional neural networks have made it possible to classify images based on an almost arbitrarily diverse array of characteristics, and previous work has been dealt with analyzing subjective beauty using convolutional neural networks (CNNs).

One major problem in the study of fractal aesthetics is dealing with the massive sample space of all possible equations that can be used to generate fractals. This large sample space makes it difficult to generate fractals that are aesthetically pleasing to humans because many equations produce fractals that visit only a few points in the X-Y plane, which we define in this paper to be sparse fractals. Sparse fractals are non-stimulating to humans and finding a way to filter these fractals out is key to presenting potentially interesting fractals to the user.

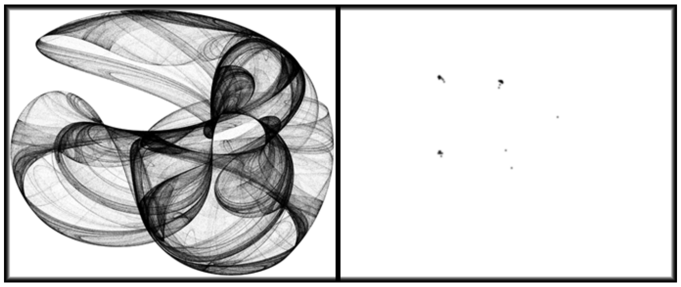


Fig. 1: Visual representation of a non-sparse fractal (left) and a sparse fractal (right).

In this paper, we combine genetic algorithms with convolutional neural networks to form a generative model of non-sparse fractals, in the hopes of optimizing towards a more consistently dense aesthetic.

II. RELATED WORK

The mathematical nature of fractals provides no measure of beauty for the resulting image, and the shape and structure of the image is often difficult to predict. However, much research has been made into guiding computational evolution through the use of human feedback. A paper written by Igor Karpov, titled Human-Assisted Neuroevolution Through Shaping, Advice and Examples[2], discusses how humans can utilize their superior intuition and analytical skills to augment machine learning and create intelligent agents. The paper explores three different methods of human-assisted machine learning: giving advice in the form of rules, demonstrating desirable behavior through example, and shaping the machine learning process through increasingly complex tasks. Karpov concludes that human-assisted evolution consistently performs better than manual scripting and unassisted evolution, though some forms of human assistance are better suited for certain tasks such as obstacle avoidance and target chasing. Using this approach, the task of measuring the beauty of fractals falls not on the computer, but the user, thereby eliminating the need for a fitness function of aesthetic value.

In [3], Miura and Gobithaasan observe the detrimental effects of Bezier, B-Spline and NURBS curves on the formulation of shape aesthetics, and propose the use of Log-Aesthetic (LA) curves as a simpler curvature alternative that can be used to design more structurally sound industrial structures with a cleaner curved aesthetic. They describe the quality of curves and surfaces using a "fairness metric" that measures a planar curve based on whether it has a continuous curvature with very few monotonic curvature pieces.

In [4], Datta et. al. build automated classiiers using SVMs and classification trees, using features extracted from crowd-sourced photo ratings from an online sharing website to form a discriminative model of appealing photographic images.

In [5], Folz et. al. use similar classifiers and apply them to a consumer-facing product that customizes photos to provide procedurally-generated aesthetic photo enhancements.

In [6], Andrej Karpathy uses a deep neural network to form a discriminative model of aesthetically-appealing selfie images, scraped from Instagram and judged based on the number of likes and user followers.

III. Genetic Representation of Fractals

Visual fractal representations are defined as a set of recursive mathematical equations outlining a sequence of points in Cartesian space. To perform optimization of a characteristic of a population using a genetic algorithm, a sample population of randomly generated individuals must first be provided. Due to the sparsity of the fractal space, a basic seeded set of equations is often desired to ensure that the initial population has a large percentage of fractals with high visual density.

$$x_{n+1} = sin(a * y_n) + c * cos(a * x_n) \tag{1}$$

$$y_{n+1} = sin(b * x_n) + d * cos(b * y_n) \tag{2}$$

The equations (1, 2) represent a family of fractals commonly known as the Clifford Attractor family. To create a new random generation, random constants are substituted for a, b, c, and d in the above form.

To provide a workable representation of these equations for use in genetic evolution, an expression tree can be used to represent operations (non-leaf nodes), constants and variables (leaves), and their mathematical relations.

IV. Fractal Evolution with Genetic Algorithms

In a genetic algorithm, a population of candidate solutions is evolved towards solutions that tend towards a feature optimization. In the fractal space, these solutions (fractals) have a set of expression trees made of operation and constant/variable nodes that can be mutated and altered to build a new generation of fractals.

Traditionally, the three evolutionary operations used in genetic programming are crossover, mutation, and cloning. By

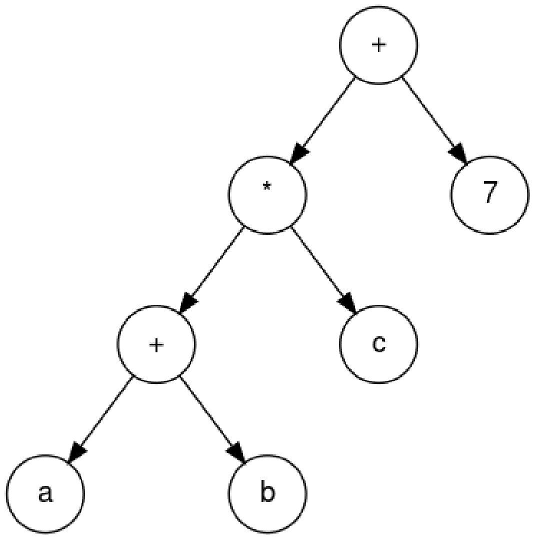


Fig. 2: An exemplary expression tree representing the equation $(a + b) * c + 7$.

keeping a record of previously generated fractals, however, cloning becomes a redundant operation as the user can simply go back to a previously generated fractal and re-evolve it. In this paper we address this problem of redundancy by presenting a new evolutionary operator in place of cloning.

An important distinction to make when evolving fractals is the difference between applying genetic operators to operations (non-leaf nodes) and constants/variables (leaf nodes). Although there is a fair amount of variety within a fractal family (i.e. a constant set of functions that are varied by varying their constants, such as the aforementioned Clifford Attractor family), applying genetic operators only to leaf nodes prevents the evolutionary process from exploring the true fractal space, containing it within a single fractal family. As with all genetic programs, a great deal of care is necessary to balance exploration and genetic drift with convergence to a satisfactory solution.

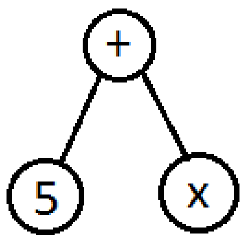
A. Crossover

Crossover is a genetic operation in which two parent fractals are combined to produce child solutions. In the fractal space, two fractals perform crossover by randomly picking one node from each of the fractals and swapping them. Within the evolutionary program built for our experiments, one of these is chosen at random, while the other is discarded.

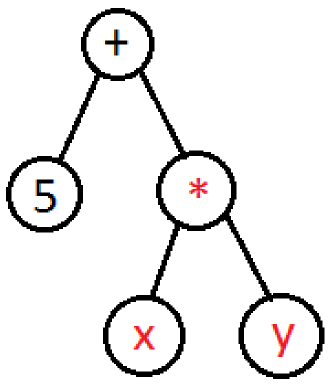
B. Leaf Mutation

Mutation is a genetic operation in which a single parent fractal has one of its gene values altered from its initial state. In this paper, we distinguish between two types of mutation: leaf mutation and a modified type of mutation performed on non-leaf nodes: mutation by insertion.

Leaf mutation is used as a way of limiting genetic drift and inconsistency by taking an existing satisfactory fractal



(a) Before Insertion.



(b) After Insertion.

Fig. 3: Mutation by Insertion. The (x) node in Figure 2a is replaced by the (x*y) tree in Figure 2b.

and slightly fuzzing its constants, resulting in a very similar visual representation.

C. *Mutation by Insertion*

Since the program supports switching back to previous generations, cloning is redundant as users could simply go back to previous generations and re-generate the next generation of fractals. For this reason, a new operation called insertion was implemented.. Insertion is a modification of the generic mutation operator that alters the expression tree for each fractal equation by picking a random leaf node and substituting a small, randomly generated expression tree for that leaf node. This randomly generated expression tree thus becomes a subtree in the original expression tree. The generated expression tree is formed to be probabilistically similar to the tree that it is replacing, to minimize the change in expression structure.

A randomly generated expression tree is created by randomly generating a single node that is either an operator, variable, or constant. If the node is an operator, then that node must have child nodes to perform the operation on. In this case children are randomly generated for that node by applying the same procedure recursively. This process continues until all leaf nodes of the expression tree are either variables or constants, i.e. they take no parameters.

One of the main benefits observed from implementing this operator is that new operators are able to appear in a fractals equation. As stated earlier in the paper, our program uses the Clifford Attractor general form when creating the first generation of fractals. This general form only uses four operators: $sin()$, $cos()$, $+$, and $*$. Thus, crossover, mutation, and cloning of fractals will only yield fractals with a combination of these four operators, as mutation is not easily applicable across the discrete domain of function structure. By using insertion, however, we are able to define a larger set of operators that can appear in the fractal equations and thus allow for more variety in the fractals.

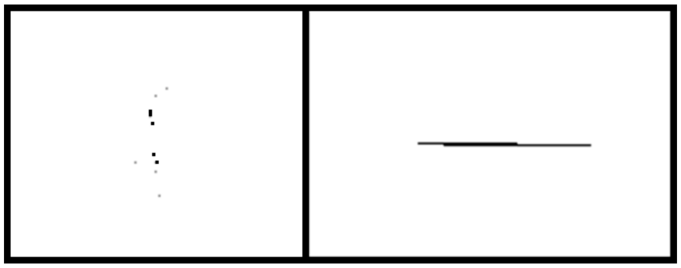


Fig. 4: Two sparse fractals. Scattered points (left) and highly-linear fractals (right) are strong indicators of fractal sparsity.

V. A Genetic Fitness Function: Fractal Sparsity

A major problem encountered when randomly evolving fractals is dealing with the large number of sparse fractals that appear in each generation. Here, the definition of a sparse fractal is one made aesthetically unappealing due to its lack of density and form. In quantitative terms, the sparsity of a fractal can be defined by one or both of the following main features:

1) An insufficiently large quantity of distinct points in Cartesian integer space.
2) A lack of non-linear figures (e.g. curves).

From the above criteria, a method was developed in order to quantitatively rank fractals as being sparse (”bad”) and non-sparse (”good”). The first approach was to use the ratio of black to white pixels in the saved 640x360 image to determine how much of the image was covered by the fractal. However, at some point, the aesthetic appeal of a fractal decreases as its visual density increases. For instance, a large black square is dense, yet visually unappealing to most humans.

The final method devised was able to ensure both criteria were met with a high rate of success and did not require much computation. Instead of looking at the individual pixels in the image of a fractal, the program examines the image file itself, specifically at how large the image file is in bytes. To understand this methodology, we must describe the format of the images created by the program.

The fractal visualization program uses the PNG image format for saving images of the generated fractals. PNG is a lossless image format, meaning no data is lost when saving the image. In contrast, formats like JPG sacrifice data quality for the sake of smaller image sizes. Thus, PNG encoders can only reduce the size of the output file through lossless compression techniques. The specific compression algorithm used by PNG encoders is called DEFLATE, the same algorithm used by ZIP archives. One such technique is to find large regions of one color and compress that region to a single block of information in the output file. This technique is very effective at reducing the sizes of images that are composed only of drawn lines because the encoder can compress each of the lines by using LZ77 compression, a method that looks for and replaces repeated sequences of data. This compression technique also highly

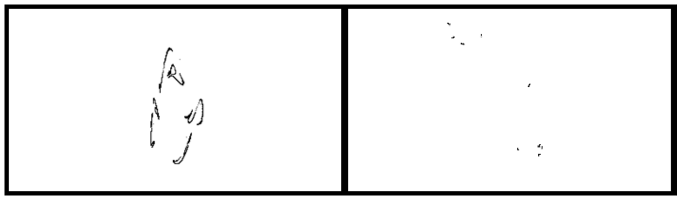


Fig. 5: Two rendered fractals. The non-sparse image on the left is represented in 2.1 kilobytes, while the image on the right is 0.68 kilobytes.

compresses images with very few points that are different in color than the background because only a few points need tobe individually saved in the output. Thus, PNG encoders effectively compress sparse fractal images. Exploiting this fact, we are able to define a size threshold for the PNG files of sparse fractals. For fractals rendered at a resolution of 640x360 pixels we defined the sparse/non-sparse cutoff to be approximately 1 kilobyte, and experiment with the deviation from this range to determine especially ”good” and ”bad” fractals for neural network tuning.

VI. Learning a Generative Model with Convolutional Neural Networks

The evolutionary program is split into two parts. The Java Swing frontend displays a range of fractal images on-screen for the user to interact with, providing tools for human-guided evolution of the fractals' genetic structures, while the C++ backend utilizes OpenGL to evaluate fractal equations, generate appropriate image files, and render fractals in high definition for closer inspection. The equation for the fractal is interpreted in Java with the help of the Exp4j library, which converts infix representationsof equations into postfix notation. Using the postfix representation of the equation, an expression tree is created with operation nodes and value leaves. Representing fractals in this form allows mutation and crossover of both constants and equation forms.

For interactive training and evolution, users are provided with a 3x3 grid of fractals constituting a single generation. The first generation is seeded with Clifford Attractors of variable constant values, and the next generation is formed through reproduction between fractal parents chosen randomly from the set of user-selected fractals. If no fractals are selected, the generation is regenerated. If one or more fractals are chosen as potential parents, the next generation is created as follows:

1) The three fractals in the top row of the viewport are generated using crossover, whereby two parents are randomly selected, and randomly selected subtrees of thoseparents function trees are swapped.
2) The three fractals in the middle row of the viewport are generated using mutation, whereby a parent is randomly selected and each constant in that parents parse tree has a random chance of being mutated by a small amount.
3) The three fractals on the bottom row of the viewport are generated using insertion.

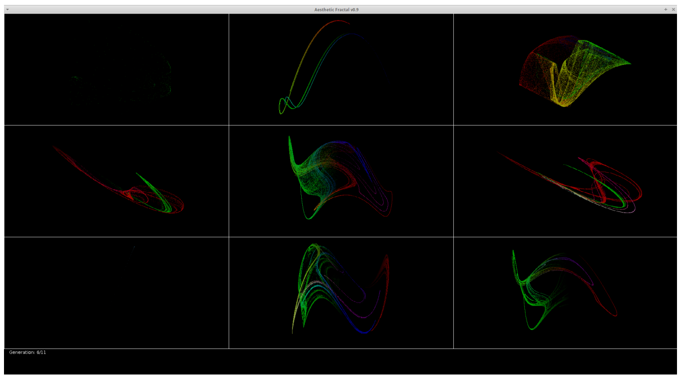


Fig. 6: The interactive evolutationary program. The RGB variables are not used in the presented tests.

Each generation is calculated and drawn in parallel, running nine threads for each generational child.

To observe quantitative results, this training process is automated using the file size metric previously described to determine positive and negative examples of good fractals.

These positive and negative examples are fed to Clarifai's CNN tuning system, which allows a basic CNN, trained on ImageNet and other competitive image knowledge bases, to be tuned to other, more specific features. Approximately 1000 total examples, half positive, half negative, were provided for training in each experimental case. The final Clarifai model was then used as to replace the sparsity metric in a new automated evolutionary process through 1000 iterations. The average sparsity for each generation was recorded and plotted in Fig. 7-9.

VII. Evaluation

To test the ability of our neural network, combined with our evolutionary algorithm, to optimize towards dense fractals, we first use our filesize sparsity metric to generate 500 positive and 500 negative fractal samples. We consider positive fractals to be those at least 1.3 kilobytes, and negative fractals to be below 0.7 kilobytes. After training our network, we can then use it to graph the trends of average sparsity using our hard metric and the neural network across 1000 additional generations.

In figures 7 and 8, it is clear that the neural network shows the same trend across 1000 generations as the quantitative file size metric. There is a signiicant dip in average quality for the first two hundred generations before rising sharply between generations 200-300, at which point it mainly stabilizes around an average file size of 1.45. The dip in quality can possibly be explained by crossover and mutation, which allows the evolutationary algorithm to explore the wide fractal space by rapidly modifying their expression structures dramatically instead of converging to better fractals. However, there is a 50-node cap on the height of an expression tree, and the steep increase across generations 200-300 can be explained by this cap being reached across all of the fractal expressions, reducing the exploratory step from generation to generation and allowing fractals to converge more rapidly

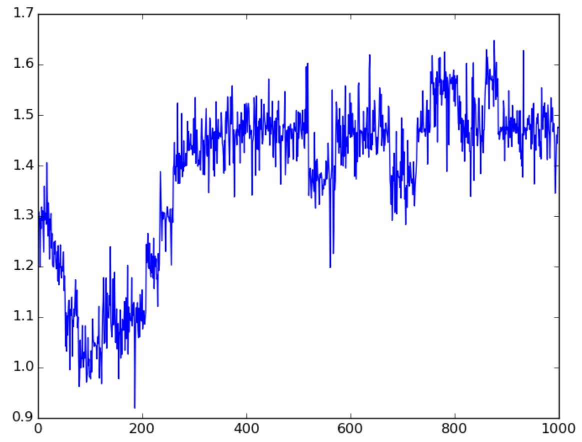


Fig. 7: The average image file size (kb) across 1,000 generations using the generative CNN model.

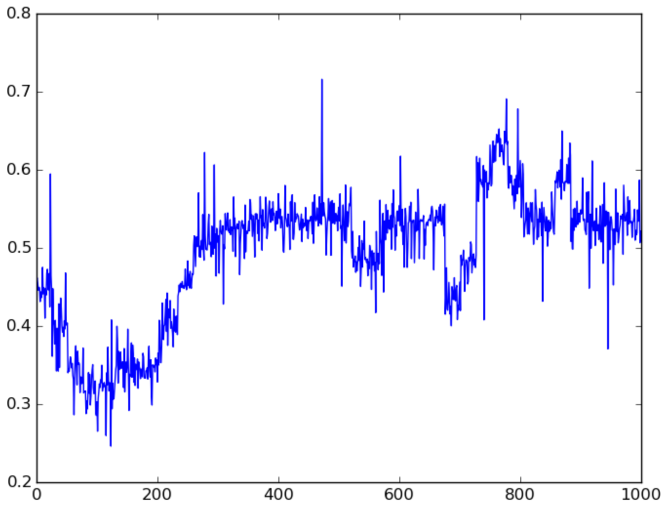


Fig. 8: The average neural network score across 1,000 generations using the generative CNN model.

at this point in time due to more consistent expression structures. The smaller dips and peaks after generation 300 can be explained as rarer generations in which cross-over and insertion result in a large population of "bad" or "good" fractals in a generation, intuitively poisoning the well or allowing the program to overcome a local optimum and find a better peak.

In figure 9, we can observe the number of negative fractals across 1,000 neural-network-generated fractal populations. Following the trend seen in the previous two charts, we can observe the number of negative fractals start around 2, increase to around 3 until generation 300, then drop down and vary between 0-2 per generation, occasionally reaching 3 and rarely reaching above it.

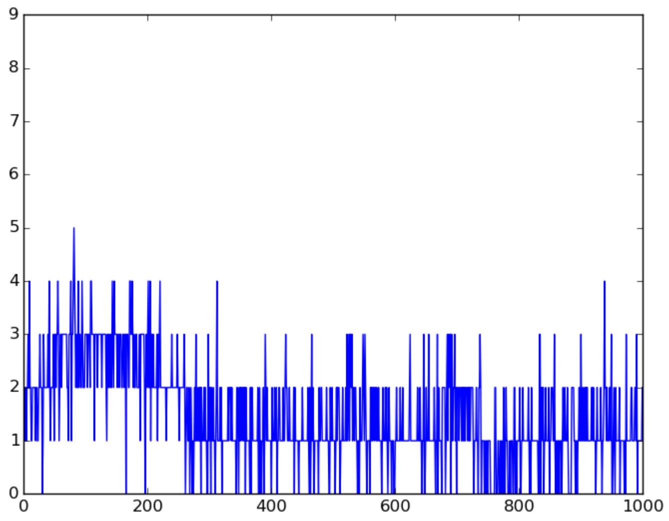


Fig. 9: The number of sparse fractals per generation, as determined by image filesize under 1kb, across 1,000 generations using the generative CNN model.

VIII. Future Work

In the future, more detailed analysis is necessary to explain the results seen. File size and neural network confidence scores should be observed based on the type of genetic operation used to generate fractals. Parameters such as file size and confidence score cutoffs should be experimented with to observe how they change the general trend of evolution, if at all, to be more or less selective. The discriminative neural network model should also be observed in regards to the known basis of positive and negative fractals to see if there is a strong correlation between the model's confidence scores and a fractal image's file size.

The interactive program is general enough to allow extra variables to be removed or added, and it would be interesting to experiment with neural-network guided evolution using a more subjective, human-guided fitness function that involves the use of RGB color expressions defined based on the XYZ variables. Another, more complex and reliable fitness function related to the curvature of the fractal would likely provide additional support for the evolutationary program to optimize towards a specific curved or sharp aesthetic.

A more subjective experiment using real human-guided training can reveal insights into how well a convolutional neural network can deal with more complex, implicit aesthetics.

IX. Discussion and Conclusion

Machine learning and Learning from Demonstration (LfD) techniques have often been used to automate control tasks with quantifiable evaluations. However, as humans, personal preferences, emotions, and aesthetics often play a role in how we perceive the world, and how we perceive art. The work presented in this paper shows promise that robots and computers can learn to understand, implicitly, the quantification of aesthetic values and, perhaps, to learn how to create art for the pleasure of human beings. Though further work is desired to test the combination of genetic evolution and neural networks in the realm of more complex, implicit humanized aesthetics – ones that cannot be easily modeled or quantified by hand – there is promise that, on some level, computers can learn the aesthetics that make art come to life.

REFERENCES

[1] ”fractal, n.1.” OED Online. Oxford University Press, September 2015. Web. 28 November 2015.

[2] Igor Karpov, Vinod Valsalam, Risto Miikkulainen. ”Human-Assisted Neuroevolution Through Shaping, Advice and Examples,” In Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO 2011), Dublin, Ireland, July 2011.

[3] K.T. Miura, R.U. Gobithaasan. ”Aesthetic curves and surfaces in computer aided geometric design,” Int. J. Autom. Technol., 8 (3) (2014), pp. 304316.

[4] Datta, Ritendra, Dhiraj Joshi, Jia Li, and James Z. Wang. ”Studying Aesthetics in Photographic Images Using a Computational Approach.” Computer Vision ECCV 2006 Lecture Notes in Computer Science (2006): 288-301. Web.

[5] Joachim Folz, Christian Schulze, Damian Borth, and Andreas Dengel. 2015. Aesthetic Photo Enhancement using Machine Learning and Case-Based Reasoning. In Proceedings of the 1st International Workshop on Affect & Sentiment in Multimedia (ASM ’15). ACM, New York, NY, USA, 27-32.

[6] Karpathy, Andrej. ”What a Deep Neural Network Thinks about Your #selfie.” What a Deep Neural Network Thinks about Your #selfie. N.p., 25 Oct. 2015. Web. 02 Dec. 2015.