Full Citation: Nguyen A, Yosinski J, Clune J. *Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images*. In Computer Vision and Pattern Recognition (CVPR '15), IEEE, 2015.

Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images

Anh Nguyen
University of Wyoming
anguyen8@uwyo.edu

Jason Yosinski
Cornell University
yosinski@cs.cornell.edu

Jeff Clune
University of Wyoming
jeffclune@uwyo.edu

Abstract

Deep neural networks (DNNs) have recently been achieving state-of-the-art performance on a variety of pattern-recognition tasks, most notably visual classification problems. Given that DNNs are now able to classify objects in images with near-human-level performance, questions naturally arise as to what differences remain between computer and human vision. A recent study [26] revealed that changing an image (e.g. of a lion) in a way imperceptible to humans can cause a DNN to label the image as something else entirely (e.g. mislabeling a lion a library). Here we show a related result: it is easy to produce images that are completely unrecognizable to humans, but that state-of-the-art DNNs believe to be recognizable objects with 99.99% confidence (e.g. labeling with certainty that white noise static is a lion). Specifically, we take convolutional neural networks trained to perform well on either the ImageNet or MNIST datasets and then find images with evolutionary algorithms or gradient ascent that DNNs label with high confidence as belonging to each dataset class. It is possible to produce images totally unrecognizable to human eyes that DNNs believe with near certainty are familiar objects. Our results shed light on interesting differences between human vision and current DNNs, and raise questions about the generality of DNN computer vision.

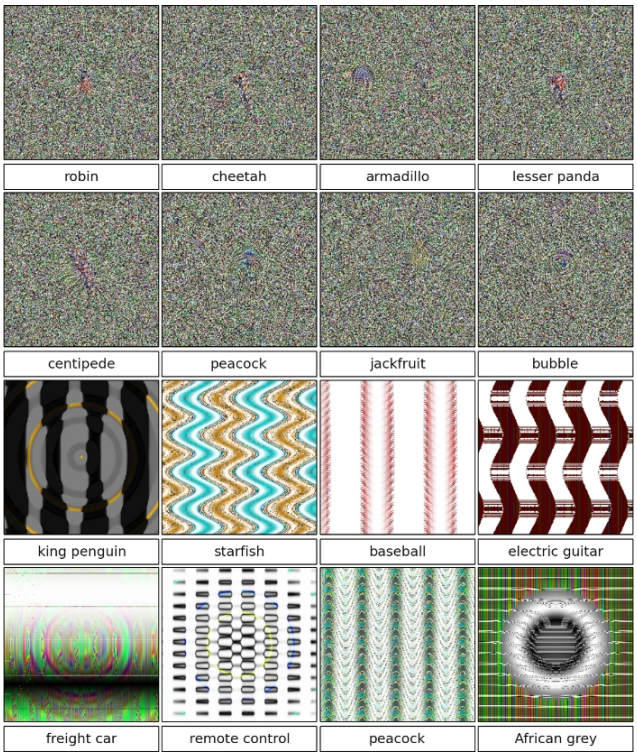

Figure 1. Evolved images that are unrecognizable to humans, but that state-of-the-art DNNs trained on ImageNet believe with $\geq$ 99.6% certainty to be a familiar object. This result highlights differences between how DNNs and humans recognize objects. Images are either directly (*top*) or indirectly (*bottom*) encoded.

1. Introduction

Deep neural networks (DNNs) learn hierarchical layers of representation from sensory input in order to perform pattern recognition [1, 13]. Recently, these deep architectures have demonstrated impressive, state-of-the-art, and sometimes human-competitive results on many pattern recognition tasks, especially vision classification problems [15, 5, 27, 16]. Given the near-human ability of DNNs to classify visual objects, questions arise as to what differences remain between computer and human vision.

A recent study revealed a major difference between DNN and human vision [26]. Changing an image, originally correctly classified (e.g. as a lion), in a way imperceptible to human eyes, can cause a DNN to label the image as something else entirely (e.g. mislabeling a lion a library).

In this paper, we show another way that DNN and human vision differ: It is easy to produce images that are completely unrecognizable to humans (Fig. 1), but that state-of-the-art DNNs believe to be recognizable objects with over 99% confidence (e.g. labeling with certainty that TV static

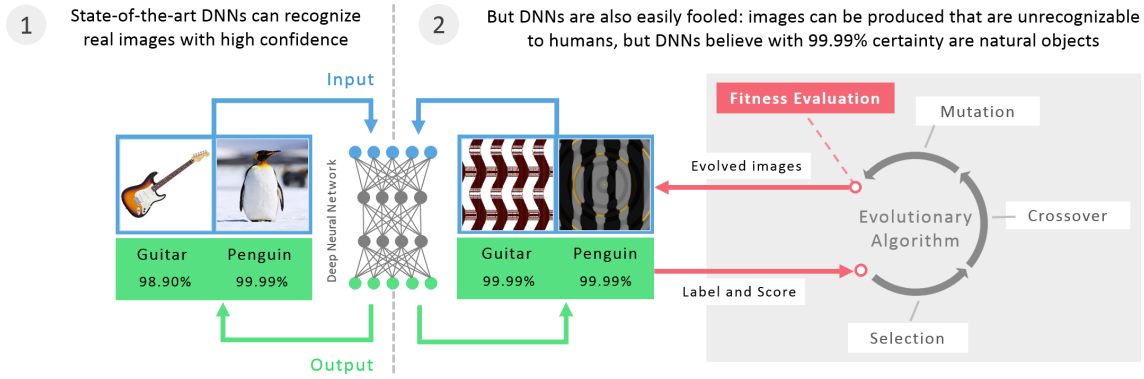

Figure 2. Although state-of-the-art deep neural networks can increasingly recognize natural images (*left panel*), they also are easily fooled into declaring with near-certainty that unrecognizable images are familiar objects (*center*). Images that fool DNNs are produced by evolutionary algorithms (*right panel*) that optimize images to generate high-confidence DNN predictions for each class in the dataset the DNN is trained on (here, ImageNet).

is a motorcycle). Specifically, we use evolutionary algorithms or gradient ascent to generate images that are given high prediction scores by convolutional neural networks (convnets) [15, 17]. These DNN models have been shown to perform well on both the ImageNet [8] and MNIST [18] datasets. We also find that, for MNIST DNNs, it is not easy to prevent the DNNs from being fooled by retraining them with fooling images labeled as such. While retrained DNNs learn to classify the negative examples as fooling images, a new batch of fooling images can be produced that fool these new networks, even after many retraining iterations.

Our findings shed light on current differences between human vision and DNN-based computer vision. They also raise questions about how DNN networks perform in general across different types of images than the ones they have been trained and traditionally tested on.

2. Methods

2.1. Deep neural network models

To test whether DNNs might give false positives for unrecognizable images, we need a DNN trained to near state-of-the-art performance. We choose the well-known "AlexNet" architecture from [15], which is a convnet trained on the 1.3-million-image ILSVRC 2012 ImageNet dataset [8, **?**]. Specifically, we use the already-trained AlexNet DNN provided by the Caffe software package [14]. It obtains 42.6% top-1 error rate, similar to the 40.7% reported by Krizhevsky 2012 [15]. While the Caffe-provided DNN has some small differences from Krizhevsky 2012 [15], we do not believe our results would be qualitatively changed by small architectural and optimization differences or their resulting small performance improvements. Similarly, while recent papers have improved upon Krizhevsky 2012, those differences are unlikely to change our results. We chose AlexNet because it is widely known and a trained DNN similar to it is publicly available. In this paper, we refer to this model as "ImageNet DNN".

To test that our results hold for other DNN architectures and datasets, we also conduct experiments with the Caffe-provided LeNet model [17] trained on the MNIST dataset [18]. The Caffe version has a minor difference from the original architecture in [17] in that its neural activation functions are rectified linear units (ReLUs) [20] instead of sigmoids. This model obtains 0.94% error rate, similar to the 0.8% of LeNet-5 [17]. In this paper, we refer to this model as "MNIST DNN".

2.2. Generating images with evolution

The novel images we test DNNs on are produced by evolutionary algorithms (EAs) [11]. EAs are optimization algorithms inspired by Darwinian evolution. They contain a population of "organisms" (here, images) that alternately face selection (keeping the best) and then random perturbation (mutation and/or crossover). Which organisms are selected depends on the *fitness function*, which in these experiments is the highest prediction value a DNN makes for that image belonging to a class (Fig. 2).

Traditional EAs optimize solutions to perform well on one objective, or on all of a small set of objectives [11] (e.g. evolving images to match a single ImageNet class). We instead use a new algorithm called the multi-dimensional archive of phenotypic elites MAP-Elites [4], which enables us to simultaneously evolve a population that contains individuals that score well on many classes (e.g. all 1000 ImageNet classes). Our results are unaffected by using the more computationally efficient MAP-Elites over single-target evolution (data not shown). MAP-Elites works by keeping the best individual found so far for each objective. Each iteration, it chooses a random organism from the population, mutates it randomly, and replaces the current champion for any objective if the new individual has higher fit-

ness on that objective. Here, fitness is determined by showing the image to the DNN; if the image generates a higher prediction score for any class than has been seen before, the newly generated individual becomes the champion in the archive for that class.

We test EAs with two different *encodings* [25, 3], meaning how an image is represented as a genome. The first has a *direct encoding*, which has one grayscale integer for each of 28×28 pixels for MNIST, and three integers (H, S, V) for each of 256×256 pixels for ImageNet. Each pixel value is initialized with uniform random noise within the $[0, 255]$ range. Those numbers are independently mutated; first by determining which numbers are mutated, via a rate that starts at 0.1 (each number has a 10% chance of being chosen to be mutated) and drops by half every 1000 generations. The numbers chosen to be mutated are then altered via the polynomial mutation operator [6] with a fixed mutation strength of 15. The second EA has an *indirect encoding*, which is more likely to produce *regular* images, meaning images that contain compressible patterns (e.g. symmetry and repetition) [19]. Indirectly encoded images tend to be regular because elements in the genome can affect multiple parts of the image [24]. Specifically, the indirect encoding here is a compositional pattern-producing network (CPPN), which can evolve complex, regular images that resemble natural and man-made objects [21, 24].

Importantly, images evolved with CPPNs can be recognized by DNNs (Fig. 3), providing an existence proof that a CPPN-encoded EA can produce images that both humans and DNNs can recognize. These images were produced on PicBreeder.org [21], a website where users manually select images they like, which become the parents of the next generation; the users thus serve as the fitness function driving an evolutionary algorithm.

CPPNs are similar to artificial neural networks (ANNs). A CPPN takes in the (x, y) position of a pixel as input, and outputs a grayscale value (MNIST) or tuple of HSV color values (ImageNet) for that pixel. Like a neural network, the function the CPPN computes depends on the number of neurons in the CPPN, how they are connected, and the weights between neurons. Each CPPN node can be one of a set of activation functions (here: sine, sigmoid, Gaussian and linear), which can provide geometric regularities to the image. For example, passing the x input into a Gaussian function will provide left-right symmetry, and passing the y input into a sine function provides top-bottom repetition. Evolution determines the topology, weights, and activation functions of each CPPN network in the population.

As is custom, and was done for the images in Fig. 3, CPPN networks start with no hidden nodes, and nodes are added over time, encouraging evolution to first search for simple, regular images before adding complexity [23]. All of our code and parameters are available upon request.

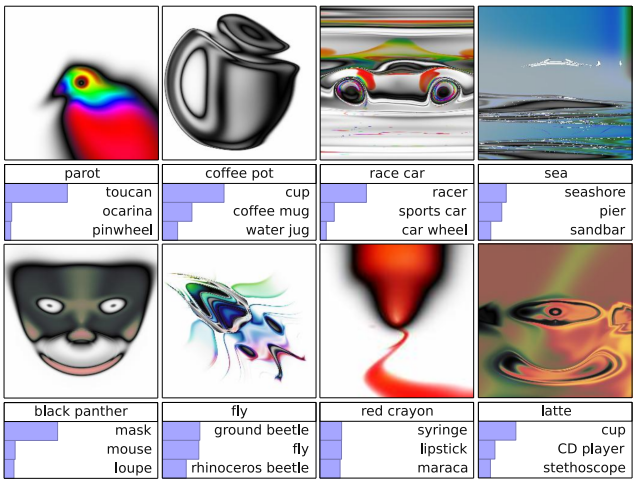

Figure 3. Evolved, CPPN-encoded images produced with humans performing selection on PicBreeder.org. Human image breeders named each object (centered text). Blue bars show the top three classifications made by a DNN trained on ImageNet (size indicates confidence). Often the first classification relates to the human breeder's label, showing that CPPN-encoded evolution can produce images that humans and DNNs can recognize.

3. Results

3.1. Evolving irregular images to match MNIST

We first evolve directly encoded images to be confidently declared by LeNet to be digits 0 thru 9 (recall that LeNet is trained to recognize digits from the MNIST dataset). Multiple, independent runs of evolution repeatedly produce images that LeNet believes with 99.99% confidence to be digits, but are unrecognizable as such (Fig. 4). In less than 50 generations, each run of evolution repeatedly produces unrecognizable images of each digit type classified by LeNet with $\geq$ 99.99% confidence. By 200 generations, median confidence is 99.99%. Given the DNN's near-certainty, one might expect these images to resemble handwritten digits. On the contrary, the generated images look nothing like the handwritten digits in the MNIST dataset.

3.2. Evolving regular images to match MNIST

Because CPPN encodings can evolve recognizable images (Fig. 3), we tested whether this more capable, regular encoding might produce more recognizable images than the irregular white-noise static of the direct encoding. The result, while containing more strokes and other regularities, still led to LeNet labeling unrecognizable images as digits with 99.99% confidence (Fig. 5) after only a few generations. By 200 generations, median confidence is 99.99%.

Certain patterns repeatedly evolve in some digit classes that appear indicative of that digit (Fig. 5). Images classified as a 1 tend to have vertical bars, while images classified as a 2 tend to have a horizontal bar in the lower half

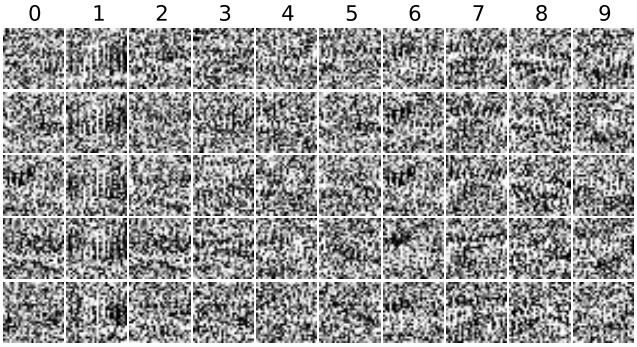

Figure 4. Directly encoded, thus irregular, images that LeNet believes with 99.99% confidence are digits 0-9. Each column is a digit class, and each row is the result after 200 generations of a randomly selected, independent run of evolution.

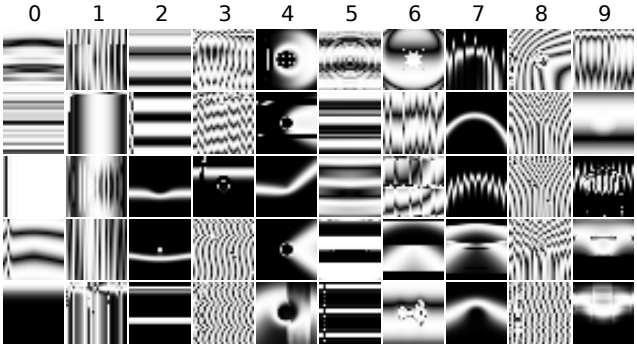

Figure 5. Indirectly encoded, thus regular, images that LeNet believes with 99.99% confidence are digits 0-9. The column and row descriptions are the same as for Fig. 4.

of the image. Qualitatively similar discriminative features are observed in 50 other runs as well (supplementary material). This result suggests that the EA exploits specific discriminative features corresponding to the handwritten digits learned by the MNIST DNN.

3.3. Evolving irregular images to match ImageNet

We hypothesized that the MNIST DNN might be easily fooled because it is trained on a small dataset that could allow for overfitting (MNIST has only 60,000 training images). To test this hypothesis that a larger dataset might prevent the pathology, we evolved directly encoded images to be classified confidently by a convolutional DNN [15] trained on the ImageNet 2012 dataset, which has 1.3 million natural images in 1000 classes [7]. Confidence scores for images were averaged over 10 crops (1 center, 4 corners and 5 mirrors) of size 227×227.

The directly encoded EA was less successful at producing high-confidence images in this case. Even after 20,000 generations, evolution failed to produce high-confidence images for many categories (Fig. 6, median confidence 21.59%). However, evolution did manage to produce images for 45 classes that are classified with $\geq 99\%$ confidence to be natural images (Fig. 1). While in some cases one might discern features of the target class in the image if told the class, humans without such priming would not recognize the image as belonging to that class.

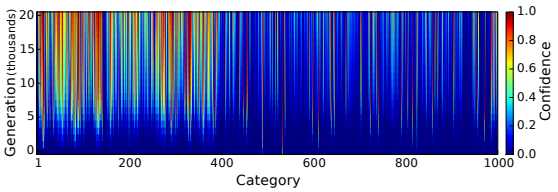

Figure 6. Median confidence scores from 5 runs of directly encoded, evolved images for all 1000 ImageNet classes. Though rare, evolution can produce images that the DNN believes with over 99% confidence to be in a natural, ImageNet class.

3.4. Evolving regular images to match ImageNet

Once again, we test whether the CPPN encoding, which has previously evolved images that both humans and DNNs recognize similarly (Fig. 3), might produce more recognizable images than the direct encoding. The hypothesis is that the larger ImageNet dataset and more powerful DNN architecture may interact with the CPPN encoding to finally produce recognizable images.

In five independent runs, evolution produces many images with DNN confidence scores of 99.99% or above, but that are unrecognizable as members of that class (Fig. 1 bottom). After 5000 generations, the median confidence score reaches 88.11%, significantly higher than the 21.59% for the direct encoding (Fig. 12, $p < 0.0001$ via Mann-Whitney U test), which was given 4-fold more generations. High-confidence images are found in most categories (Fig. 7).

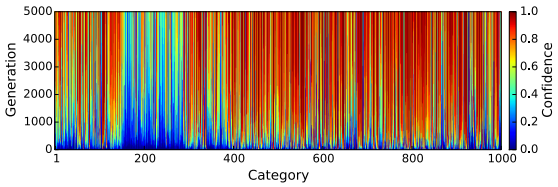

Figure 7. Median confidence scores from 5 runs of CPPN-encoded, evolved images for all 1000 ImageNet classes. Evolution can produce many images that the DNN believes with over 99% confidence to belong to ImageNet classes.

While a human not given the class labels for CPPN images would not label them as belonging to that class, the generated images do often contain some features of the target class. For example, in Fig. 1, the starfish image contains the blue of water and the orange of a starfish, the baseball has red stitching on a white background, the remote control has a grid of buttons, etc. For many of the produced images, one can begin to identify why the DNN believes the image is of that class once given the class label. This is because

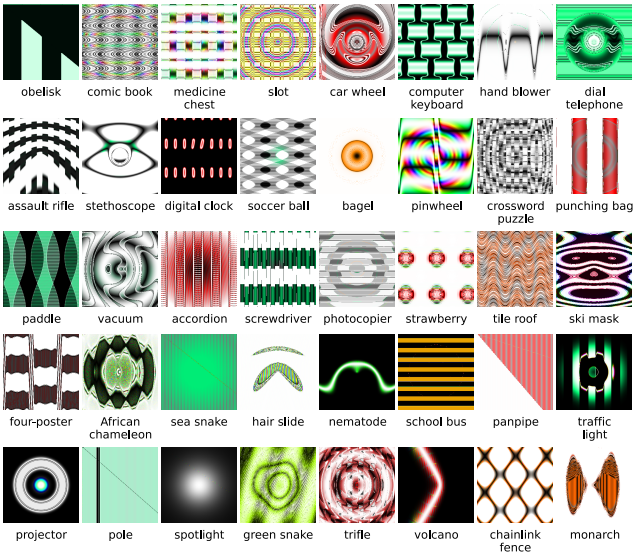

Figure 8. Evolving images to match DNN classes produces a tremendous diversity of images. Shown are images selected to showcase diversity from 5 evolutionary runs. The diversity suggests that the images are non-random, but that instead evolutions producing discriminative features of each target class. The mean DNN confidence scores for these images is 99.12%.

evolution need only to produce features that are unique to, or *discriminative* for, a class, rather than produce an image that contains all of the typical features of a class.

The pressure to create these discriminative features led to a surprising amount of diversity in the images produced (Fig. 8). That diversity is especially noteworthy for two reasons: (1) it has been shown that imperceptible changes to an image can change a DNN's class label, so it could have been the case that evolution produced very similar images for all classes that all generated high confidence scores [26], and (2) many of the images are related to each other phylogenetically. That relatedness leads evolution to produce similar images to fool closely related categories (Fig. 9). For example, one class of images receive high confidence scores for three types of lizards, and a different class of images receive high confidence scores for three types of small, fluffy dogs. Different runs of evolution, however, produce different classes of images for these related categories, revealing that there are different discriminative features per class that evolution exploits. That suggests that there are many different ways to fool the same DNN for each class.

Many of the CPPN images feature a pattern repeated many times. To test whether that repetition improves the confidence score a DNN gives an image, or whether the repetition stems solely from the fact that CPPNs tend to produce regular images [24, 3], we ablated (i.e. removed) some of the repeated elements to see if the DNN confidence score for that image drops. In many images, ablating extra copies

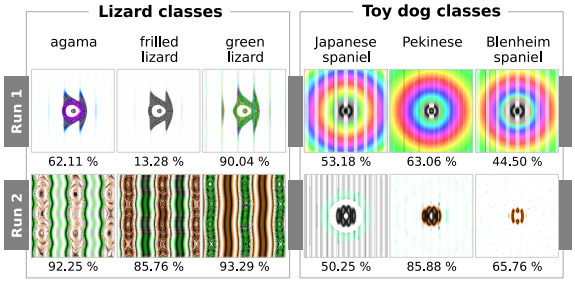

Figure 9. Images from the same evolutionary run that fool closely related classes are similar. Shown are the top images evolution generated for three classes that belong to the "lizard" parent class, and for three classes that belong to "toy dog" parent class. The top and bottom rows show images from independent runs of evolution.

of the repeated element did lead to a performance drop, albeit a small one (Fig 10), meaning that the extra copies make the DNN more confident that the image belongs to the target class. This result is in line with a previous paper [22] that produced images to maximize DNN confidence scores (discussed below in Section 3.9), which also saw the emergence of features (e.g. a fox's ears) repeated throughout an image. These results suggest that DNNs tend to learn low- and middle-level features rather than the global structure of objects. If DNNs were properly learning global structure, images should receive lower DNN confidence scores if they contain repetitions of object subcomponents that rarely appear in natural images, such as many pairs of fox ears or endless remote buttons (Fig. 1). An alternate explanation is that many natural images do in fact contain multiple copies of objects (e.g. several oboes Fig. 10).

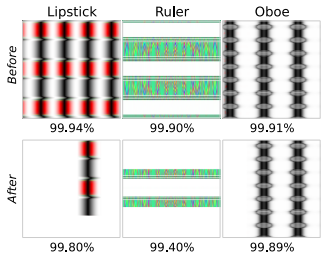

Figure 10. *Before*: CPPN-encoded images with highly regular repeated patterns. *After*: the images after being manually hand-ablated of the repeated features in such a way that the confidence scores remain close to what they were originally. The examples demonstrate that cloning a part of an image that causes high activation to other places in the image can result in even higher activation.

The low-performing band of classes in Fig. 7 (class numbers 157-286) are dogs and cats, which are overrepresented in the ImageNet dataset (i.e. there are many more classes of cats than classes of cars). One possible explanation for why images in this band receive low confidence scores is that the network is tuned to identify many specific types of dogs and

cats. Therefore, it ends up having more units dedicated to this type of image than others. In other words, the size of the dataset of cats and dogs it has been trained on is larger than for other categories, meaning it is less overfit, and thus more difficult to fool. If true, this explanation means that larger datasets are a way to ameliorate the problem of DNNs being easily fooled. An alternate, though not mutually exclusive, explanation is that, because there are more cat and dog classes, the EA had difficulty finding an image that scores high in a specific dog category (e.g. Japanese spaniel), but low in any other related categories (e.g. Blenheim spaniel), which is necessary to produce a high confidence given that the final DNN layer is softmax. This explanation suggests that datasets with more classes can help ameliorate fooling.

3.5. Do images that fool one network generalize to fooling other networks?

The results of the previous section suggest that there are discriminative features of a class of images that DNNs learn and evolution exploits. One question is whether different DNNs learn the same features for each class, or whether each trained DNN learns different discriminative features. One way to shed light on that question is to see if images that fool one DNN also fool another. To test that, we evolved CPPN-encoded images with one DNN (DNN_A) and then input them to another DNN (DNN_B), where DNN_A and DNN_B have identical architectures and training, and differ only in their randomized initializations. We performed this test for both MNIST and ImageNet DNNs.

We find that images are evolved that are given $\geq 99.99\%$ confidence scores by both DNN_A and DNN_B. Thus, some general properties of the DNNs are exploited by the CPPN-encoded EA. However, there are also images specifically fine-tuned to score high on DNN_A, but not on DNN_B. More details of the experiment and its result are in the supplementary material.

3.6. Training networks to recognize fooling images to prevent fooling

One might respond to the result that DNNs are easily fooled by saying that, while DNNs are easily fooled when images are optimized to produce high DNN confidence scores, the problem could be solved by simply changing the training regimen to include negative examples. In other words, a network could be retrained and told that the images that previously fooled it should not be considered members of any of the original classes, but instead should be recognized as a new "fooling images" class.

We tested that hypothesis with CPPN-encoded images on both MNIST and ImageNet DNNs. The process is as follows: We train DNN_1 on a dataset (e.g. ImageNet), then evolve CPPN images that produce a high confidence score for DNN_1 for the n classes in the dataset, then we take those images and add them to the dataset in a new class $n+1$; then we train DNN_2 on this enlarged "+1" dataset; (optional) we repeat the process, but put the images that evolved for DNN_2 in the $n+1$ category (a $n+2$ category is unnecessary because any images that fool a DNN are "fooling images" and can thus go in the $n+1$ category). Specifically, to represent different types of images, each iteration we add to this $n+1$ category m images randomly sampled from both the first and last generations of multiple runs of evolution that produce high confidence images for DNN_i. Each evolution run on MNIST or ImageNet produces 20 and 2000 images respectively, with half from the first generation and half from the last. Error-rates for trained DNN_i are similar to DNN_1 (supplementary material).

3.7. Training MNIST DNNs with fooling images

To make the $n+1$ class have the same number of images as other MNIST classes, the first iteration we add 6000 images to the training set (taken from 300 evolutionary runs). For each additional iteration, we add 1000 new images to the training set. The immunity of LeNet is not boosted by retraining it with fooling images as negative examples. Evolution still produces many unrecognizable images for DNN_2 with confidence scores of 99.99%. Moreover, repeating the process for 15 iterations does not help (Fig. 11), even though DNN_{15}'s overrepresented 11th "fooling image class" contains 25% of the training set images.

3.8. Training ImageNet DNNs with fooling images

The original ILSVRC 2012 training dataset was extended with a 1001st class, to which we added 9000 images that fooled DNN_1. That 7-fold increase over the 1300 images per ImageNet class is to emphasize the fooling images in training. Without this imbalance, training with negative examples did not prevent fooling; MNIST retraining did not benefit from over representing the fooling image class.

Contrary to the result in the previous section, for ImageNet models, evolution was less able to evolve high confidence images for DNN_2 than DNN_1. The median confidence score significantly decreased from 88.1% for DNN_1 to 11.7% for DNN_2 (Fig. 12, $p < 0.0001$ via Mann-Whitney U test). We suspect that ImageNet DNNs were better inoculated against being fooled than MNIST DNNs when trained with negative examples because it is easier to learn to tell CPPN images apart from natural images than it is to tell CPPN images from MNIST digits.

To see whether this DNN_2 had learned features specific to the CPPN images that fooled DNN_1, or whether DNN_2 learned features general to all CPPN images, even recognizable ones, we input recognizable CPPN images from Picbreeder.org to DNN_2. DNN_2 correctly labeled 45 of 70 (64%, top-1 prediction) PicBreeder images as CPPN images, despite having never seen CPPN images like them be-

fore. The retrained model thus learned features generic to CPPN images, helping to explain why producing new images that fool DNN_2 is more difficult.

3.9. Producing fooling images via gradient ascent

A different way to produce high confidence, yet mostly unrecognizable images is by using gradient ascent in pixel space [10, 22, 26]. We calculate the gradient of the posterior probability for a specific class — here, a softmax output unit of the DNN — with respect to the input image using backprop, and then we follow the gradient to increase a chosen unit's activation. This technique follows [22], but whereas we aim to find images that produce high confidence classifications, they sought visually recognizable "class appearance models." By employing L2-regularization, they produced images with some recognizable features of classes (e.g. dog faces, fox ears, and cup handles). However, their confidence values are not reported, so to determine the degree to which DNNs are fooled by these backpropagated images, we replicated their work (with some minor changes, see supplementary material) and found that images can be made that are also classified by DNNs with 99.99% confidence, despite them being mostly unrecognizable (Fig. 13). These optimized images reveal a third method of fooling DNNs that produces qualitatively different examples than the two evolutionary methods in this paper.

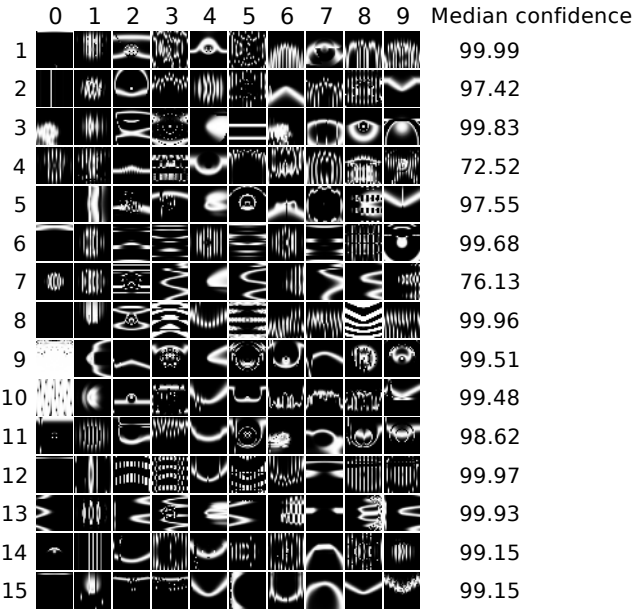

Figure 11. Training LeNet DNN_i with images that fooled LeNet DNN_1 through DNN_{i-1} does not prevent evolution from finding new fooling images for DNN_i. Columns are digits. Rows are DNN_i for $i = 1...15$. Each row shows the 10 final, evolved images from one randomly selected run of the 30 runs conducted per iteration. Medians are taken from images from all 30 runs.

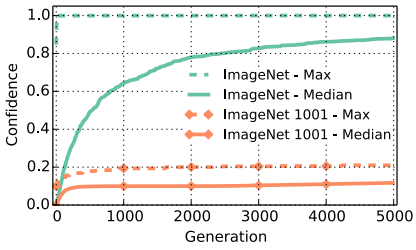

Figure 12. Training a new ImageNet DNN (DNN_2) with images that fooled a previous DNN (DNN_1) makes it significantly more difficult for evolution to produce high confidence images.

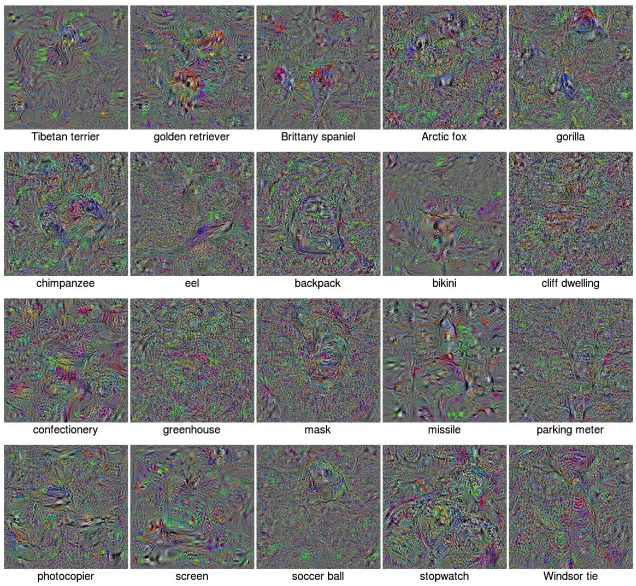

Figure 13. Images found by directly maximizing the posterior probability (softmax output) for several classes via gradient ascent, following [10, 22]. Optimization begins at the ImageNet mean (plus small Gaussian noise to break symmetry) and continues until the DNN confidence for the target class reaches 99.99%. For clarity, images are shown with the mean subtracted and with the contrast slightly enhanced. Optimization was not regularized; with L2-regularization, images are more recognizable, but have slightly lower confidence scores (see the supplementary material).

4. Discussion

Our experiments could have led to very different results. One might have expected evolution to produce *very similar*, high confidence images for all classes, given that [26] recently showed that imperceptible changes to an image can cause a DNN to switch from classifying it as class A to class B (Fig. 14). Instead, evolution produced a tremendous diversity of images (Figs. 1, 8, 10, 15). Alternately, one might have predicted that evolution would produce *recognizable* images for each class given that, at least with the CPPN encoding, recognizable images have been evolved (Fig. 3). We note that we did not set out to produce unrecognizable images that fool DNNs. Instead, we had hoped the resultant

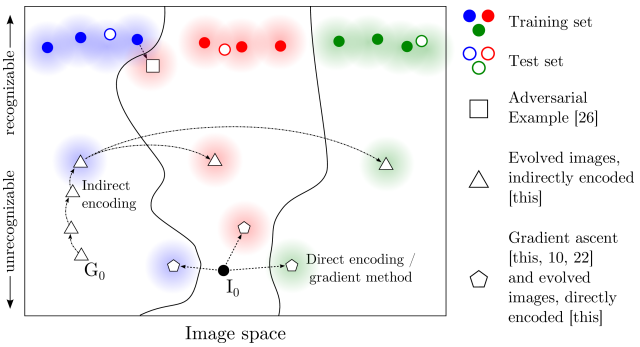

Figure 14. Interpreting our results and those of related works. (1) Szegedy et. al. [26] found that applying an imperceptible perturbation to a correctly classified natural image (blue dot) results in an image (square) that a DNN classifies as an entirely different class (crossing the decision boundary). The difference between the original image and the modified one is imperceptible to human eyes. (2) It is possible to find high-confidence images (pentagon) using our directly encoded EA or gradient ascent optimization starting from a random or blank image (I_0) [10, 22]. These images have blurry, discriminative features of the represented classes, but do not look like images in the training set. (3) We found that using indirectly encoded EAs, one can find high-confidence, regular images (triangles) that have discriminative features for a class, but are still far from the training set.

images would be recognizable. A further prediction could have been that evolution would fail to produce *high confidence* scores at all, which could happen if rugged fitness landscapes led it to become trapped in poor local optima.

In fact, none of these outcomes resulted. Instead, evolution produced high-confidence, yet unrecognizable images. Why? Our leading hypothesis is because the DNNs we experimented with are *discriminative models*, instead of *generative models*. Discriminative models create decision boundaries that partition data into classification regions. The further from the decision boundary an image is, even if far away from the natural images in the class, the higher the confidence. At a minimum, in a high-dimensional space, the area allocated by a discriminative model to a class could be very large, such that a DNN can be highly confident that a synthetic image is in that area without that image being close to the natural images in that area (see lower regions in Fig. 14).

A generative model, in contrast, builds a density model of the points in a class, thus creating a much smaller area per class. Thus, synthetic images that produce high confidence values for generative models may be closer to natural images, and thus more recognizable. Testing whether our results hold for generative models is an interesting area of future work. Performing that study was beyond the scope of this work, and may prove more fruitful to conduct once generative models can scale to the high-dimensionality of datasets like ImageNet instead of being restricted to modeling data with fewer dimensions [2].

Figure 15. Some evolved images do resemble their target class. In each pair, an evolved, CPPN-encoded image (*left*) is shown with a training set image from the target class (*right*).

In this paper we focus on the fact that there exist images that DNNs declare with near-certainty to be of a class, but are unrecognizable as such. However, it is also interesting that some generated images do resemble their target class, at least once the class label is known. Fig. 15 juxtaposes examples with natural images from the target class. Other examples include the chain-link fence, computer keyboard, digital clock, bagel, strawberry, ski mask, spotlight, and monarch butterfly of Fig. 8. In a companion paper, we explore how these successes give us hope that we can combine DNNs with evolutionary algorithms to promote more open-ended, creative search algorithms (Nguyen, Yosinski, and Clune, In prep.).

Moreover, the CPPN-encoded EA presented can be considered a novel technique to visualize some of the features learned by DNNs. The diversity of patterns generated for the same class over different runs (Fig. 9) indicates the diversity of features learned for that class. This and other tools for visualizing DNN features can help researchers understand the effects of unsupervised pre-training [9], fine-tuning [12], or transferring features from other networks [28].

One interesting implication of the fact that DNNs are easily fooled is that such false positives could be exploited wherever DNNs are deployed. For example, one can imagine a security camera that relies on face or voice recognition being compromised. Swapping white-noise for a face, fingerprints, or a voice might be especially pernicious since other humans nearby might not recognize that someone is attempting to compromise the system. Another area of concern could be image-based search engine rankings: background patterns that a visitor does not notice could fool a DNN-driven search engine into thinking a page is about an altogether different topic. The fact that DNNs are increasingly used in a wide variety of industries, including safety-critical ones such as driverless cars, raises the possibility of costly exploits via techniques that generate fooling images.

5. Conclusion

We have demonstrated that discriminative DNN models are easily fooled in that they classify many unrecognizable images with near-certainty as members of a recognizable class. Two different ways of encoding evolutionary algorithms produce two qualitatively different types of unrec-

ognizable "fooling images", and gradient ascent produces a third. That DNNs see these objects as near-perfect examples of recognizable images sheds light on remaining differences between the way DNNs and humans recognize objects, raising questions about the true generalization capabilities of DNNs and the potential for costly exploits of solutions that use DNNs.

Acknowledgments

The authors would like to thank Hod Lipson for helpful discussions and the NASA Space Technology Research Fellowship (JY) for funding. We also thank Joost Huizinga, Christopher Stanton, and Jingyu Li for helpful feedback.

References

[1] Y. Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. 1

[2] Y. Bengio, E. Thibodeau-Laufer, G. Alain, and J. Yosinski. Deep generative stochastic networks trainable by backprop. In *Proceedings of the International Conference on Machine Learning*, 2014. 8

[3] J. Clune, K. Stanley, R. Pennock, and C. Ofria. On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Evolutionary Computation*, 15(4):346–367, 2011. 3, 5

[4] A. Cully, J. Clune, and J.-B. Mouret. Robots that can adapt like natural animals. *arXiv preprint arXiv:1407.3501*, 2014. 2

[5] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012. 1

[6] K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001. 3

[7] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. Imagenet large scale visual recognition competition 2012 (ilsvrc2012), 2012. 4

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 2

[9] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010. 8

[10] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *Dept. IRO, Université de Montréal, Tech. Rep*, 2009. 7, 8

[11] D. Floreano and C. Mattiussi. *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT press, 2008. 2

[12] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504, 2006. 8

[13] G. E. Hinton. Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434, 2007. 1

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 2

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 2, 4

[16] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3361–3368. IEEE, 2011. 1

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[18] Y. LeCun and C. Cortes. The mnist database of handwritten digits, 1998. 2

[19] H. Lipson. Principles of modularity, regularity, and hierarchy for scalable systems. *Journal of Biological Physics and Chemistry*, 7(4):125, 2007. 3

[20] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010. 2

[21] J. Secretan, N. Beato, D. B. D Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley. Picbreeder: evolving pictures collaboratively online. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1759–1768. ACM, 2008. 3

[22] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 5, 7, 8

[23] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002. 3

[24] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007. 3, 5

[25] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003. 3

[26] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1, 5, 7, 8

[27] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1701–1708. IEEE, 2014. 1

[28] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3320–3328. Curran Associates, Inc., Dec. 2014. 8

Supplementary Material for Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images

A. Do images that fool one network generalize to fooling other networks?

As we wrote in the paper: "One question is whether different DNNs learn the same features for each class, or whether each trained DNN learns different discriminative features. One way to shed light on that question is to see if images that fool one DNN also fool another. To test that, we evolved CPPN-encoded images with one DNN (DNN_A) and then input them to another DNN (DNN_B), where DNN_A and DNN_B have identical architectures and training, and differ only in their randomized initializations. We performed this test for both LeNet and ImageNet DNNs." Here we show the details of this experiment and its results.

We performed this test with two MNIST [3] DNNs (MNIST$_A$ and MNIST$_B$) and two ImageNet [2] DNNs (ImageNet$_A$ and ImageNet$_B$). 300 images were produced with each MNIST DNN, and 1000 images were produced with each ImageNet DNN.

Taking images evolved to score high on DNN_A and inputting them to DNN_B (and vice versa), we find that there are many evolved images that are given the same top-1 prediction label by both DNN_A and DNN_B (Table S1a). Furthermore, among those images, many are given $\geq$ 99.99% confidence scores by both DNN_A and DNN_B (Table S1b). Thus, evolution produces patterns that are generally discriminative of a class to multiple, independently trained DNNs. On the other hand, there are still images labeled differently by DNN_A and DNN_B (Table S1a). These images are specifically fine-tuned to exploit the original DNN. We also find $\geq$ 92.18% of the images that are given the same top-1 prediction label by both networks, are given higher confidence score by the original DNN (Table S1c).

From the experiment with MNIST DNNs, we observed that images evolved to represent digit classes 9, 6, and 2 fooled both networks DNN_A and DNN_B the most. Furthermore, these images revealed distinctive patterns (Figure S1).

Dataset	ImageNet		MNIST	
	DNN_A on DNN_B images	DNN_B on DNN_A images	DNN_A on DNN_B images	DNN_B on DNN_A images
Top-1 matches	62.8	65.9	43.3	48.7
(a) Average	64.4		46.0	
Top-1 matches scoring 99%	5.0	7.2	27.3	27.3
(b) Average	6.1		27.3	
Top-1 matches scoring higher on original DNN	95.1	98.0	88.5	95.9
(c) Average	96.6		92.2	

Table S1.
Top-1 matches: The percent of images that are given the same top-1 label by both DNN_A and DNN_B.
Top-1 matches scoring 99%: The percent of images for which both DNN_A and DNN_B believe the top-1 predicted label to be the same and the two confidence scores given are both $\geq$ 99%.
Top-1 matches scoring higher: Of the images that are given the same top-1 label by both DNN_A and DNN_B, the percent that are given a higher confidence score by the original DNN than by the other, testing DNN.

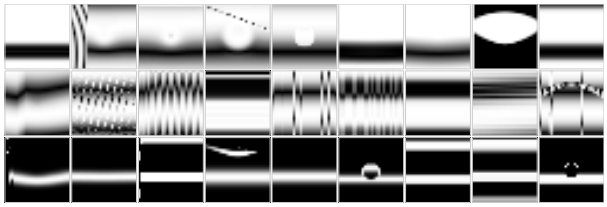

Figure S1. CPPN-encoded, evolved images which are given $\geq$ 99% confidence scores by both DNN_A and DNN_B to represent digits 9, 6, and 2. Each column represents an image produced by an independent run of evolution, yet evolution converges on a similar design, which fools not just the DNN it evolved with, but another, independently trained DNN as well.

B. Training networks to recognize fooling images to prevent fooling

As we wrote in the paper: "One might respond to the result that DNNs are easily fooled by saying that, while DNNs are easily fooled when images are optimized to pro-

duce high DNN confidence scores, the problem could be solved by simply changing the training regimen to include negative examples. In other words, a network could be re-trained and told that the images that previously fooled it should not be considered members of any of the original classes, but instead should be recognized as a new fooling images class."

We tested this hypothesis with CPPN-encoded images on both MNIST and ImageNet DNNs. The process is as follows: We train DNN_1 on a dataset (e.g. ImageNet), then evolve CPPN images that are given a high confidence score by DNN_1 for the n classes in the dataset, then we take those images and add them to the dataset in a new class $n+1$; then we train DNN_2 on this enlarged "+1" dataset; (optional) we repeat the process, but put the images that evolved for DNN_2 in the $n+1$ category (a $n+2$ category is unnecessary because any images that fool a DNN are "fooling images" and can thus go in the $n+1$ category).

Specifically, to represent different types of images, each iteration we add to this $n+1$ category m images. These images are randomly sampled from both the first and last generations of multiple runs of evolution that produce high confidence images for DNN_i. Each run of evolution on MNIST or ImageNet produces 20 or 2000 images, respectively, with half from the first generation and half from the last. As in the original experiments evolving images for MNIST, each evolution run on MNIST or ImageNet lasts for 200 or 5000 generations, respectively. These generation numbers were chosen from the previous experiments. The specific training details are presented in the following sections.

B.1. Training MNIST DNNs with fooling images

To make the $n+1$ class have the same number of images as other MNIST classes, the first iteration we add 6000 and 1000 images to the training and validation sets, respectively. For each additional iteration, we add 1000 and 100 new images to the training and validation sets (Table S2).

MNIST DNNs ($DNN_1 - DNN_{15}$) were trained on images of size 28×28, using stochastic gradient descent (SGD) with a momentum of 0.9. Each iteration of SGD used a batch size of 64, and a multiplicative weight decay of 0.0005. The learning rate started at 0.01, and reduced every iteration by an *inverse* learning rate policy (defined in Caffe [1]) with power $= 0.75$ and gamma $= 0.0001$. $DNN_2 - DNN_{15}$ obtained similar error rates to the 0.94% of DNN_1 trained on the original MNIST (Table S2).

Since evolution still produced many unrecognizable images for DNN_2 with confidence scores of 99.99%, we repeated the process for 15 iterations (Table S2). However, the retraining does not help, even though DNN_{15}'s over-represented 11th "fooling image class" contains ~25% of the training set images.

i	Error	MNIST Error	Train	Val	Score
1	0.94	0.94	60000	10000	99.99
2	1.02	0.87	66000	11000	97.42
3	0.92	0.87	67000	11100	99.83
4	0.89	0.83	68000	11200	72.52
5	0.90	0.96	69000	11300	97.55
6	0.89	0.99	70000	11400	99.68
7	0.86	0.98	71000	11500	76.13
8	0.91	1.01	72000	11600	99.96
9	0.90	0.86	73000	11700	99.51
10	0.84	0.94	74000	11800	99.48
11	0.80	0.93	75000	11900	98.62
12	0.82	0.98	76000	12000	99.97
13	0.75	0.90	77000	12100	99.93
14	0.80	0.96	78000	12200	99.15
15	0.79	0.95	79000	12300	99.15

Table S2. Details of 15 training iterations of MNIST DNNs. DNN_1 is the model trained on the original MNIST dataset without CPPN images. $DNN_2 - DNN_{15}$ are models trained on the extended dataset with CPPN images added.
Error: The error (%) on the validation set (with CPPN images added).
MNIST Error: The error (%) on the original MNIST validation set (10,000 images).
Train: The number of images in the training set.
Val: The number of images in the validation set.
Score: The median confidence scores (%) of images produced by evolution for that iteration. These numbers are also provided in the paper.

B.2. Training ImageNet DNNs with fooling images

The original ILSVRC 2012 training dataset was extended with a 1001st class, to which we added 9000 images and 2000 images that fooled DNN_1 to the training and validation sets, respectively. That ~7-fold increase over the ~1300 training images per ImageNet class is to emphasize the fooling images in training. Without this imbalance, training with negative examples did not prevent fooling; retrained MNIST DNNs did not benefit from this strategy of over representing the fooling image class (data not shown).

The images produced by DNN_1 are of size 256×256 but cropped to 227×227 for training. DNN_2 was trained using SGD with a momentum of 0.9. Each iteration of SGD used a batch size of 256, and a multiplicative weight decay of 0.0005. The learning rate started at 0.01, and dropped by a factor of 10 every 100,000 iterations. Training stopped after 450,000 iterations. The whole training procedure took ~10 days on an Nvidia K20 GPU.

Training DNN_2 on ImageNet yielded a top-1 error rate of 41.0%, slightly better than the 42.6% for DNN_1: we hypothesize the improved error rate is because the 1001st CPPN image class is easier than the other 1000 classes, because it represents a different *style* of images, making it

easier to classify them. Supporting this hypothesis is the fact that DNN_2 obtained a top-1 error rate of 42.6% when tested on the original ILSVRC 2012 validation set.

In contrast to the result in the previous section, for ImageNet models, evolution was less able to evolve high confidence images for DNN_2 compared to the high confidences evolution produced for DNN_1. The median confidence score significantly decreased from 88.1% for DNN_1 to 11.7% for DNN_2 ($p < 0.0001$ via Mann-Whitney U test).

C. Evolving regular images to match MNIST

As we wrote in the paper: "Because CPPN encodings can evolve recognizable images, we tested whether this more capable, regular encoding might produce more recognizable images than the irregular white-noise static of the direct encoding. The result, while containing more strokes and other regularities, still led to LeNet labeling unrecognizable images as digits with 99.99% confidence after only a few generations. By 200 generations, median confidence is 99.99%.". Here we show 10 images $\times 50$ runs $= 500$ images produced by the CPPN-encoded EA that an MNIST DNN believes with 99.99% to be handwritten digits (Fig. S2).

Looking at these images produced by 50 independent runs of evolution, one can observe that images classified as a 1 tend to have vertical bars. Images classified as a 2 tend to have a horizontal bar in the lower half of the image. Moreover, since an 8 can be drawn by mirroring a 3 horizontally, the DNN may have learned some common features from these two classes from the training set. Evolution repeatedly produces similar patterns for class 3 and class 8.

D. Gradient ascent with regularization

In the paper we showed images produced by direct gradient ascent to maximize the posterior probability (softmax output) for 20 example classes. Directly optimizing this objective quickly produces confidence over 99.99% for unrecognizable images. By adding different types of regularization, we can also produce more recognizable images. We tried three types of regularization, highlighted in the Figs. S3, S4, and S5.

Fig. S3 shows L2-regularization, implemented as a weight decay each step. At each step of the optimization, the current mean-subtracted image X is multiplied by a constant $1 - \gamma$ for small γ. Fig. S3 shows $\gamma = 0.01$.

Fig. S4 shows weight decay (now with $\gamma = 0.001$) plus two other types of regularization. The first additional regularization is a small blurring operator applied each step to bias the search toward images with less high frequency information and more low frequency information. This was implemented via a Gaussian blur with radius 0.3 after every gradient step. The second additional regularization was a pseudo-L1-regularization in which the (R, G, B) pixels with norms lower than the 20^{th} percentile were set to 0. This tended to produce slightly sparser images.

Finally, Fig. S5 shows a lower learning rate with the same weight decay and slightly more aggressive blurring. Because the operations of weight decay and blurring do not depend on the learning rate, this produces an objective containing far more regularization. As a result, many of the classes never achieve 99%, but the visualizations are of a different quality and, in some cases, more clear.

All images generated in this manner are optimized by starting at the ImageNet mean plus a small amount of Gaussian noise to break symmetry and then following the gradient. The noise has a standard deviation of $1/255$ along each dimension, where dimensions have been scaled to fall into the range $[0, 1]$. Because of this random initialization, the final image produced depends on the random draw of Gaussian noise. Fig. S6 and Fig. S7 show the variety of images that may be produced by taking different random draws of this initial noise.

Supplementary References

[1] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 2

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

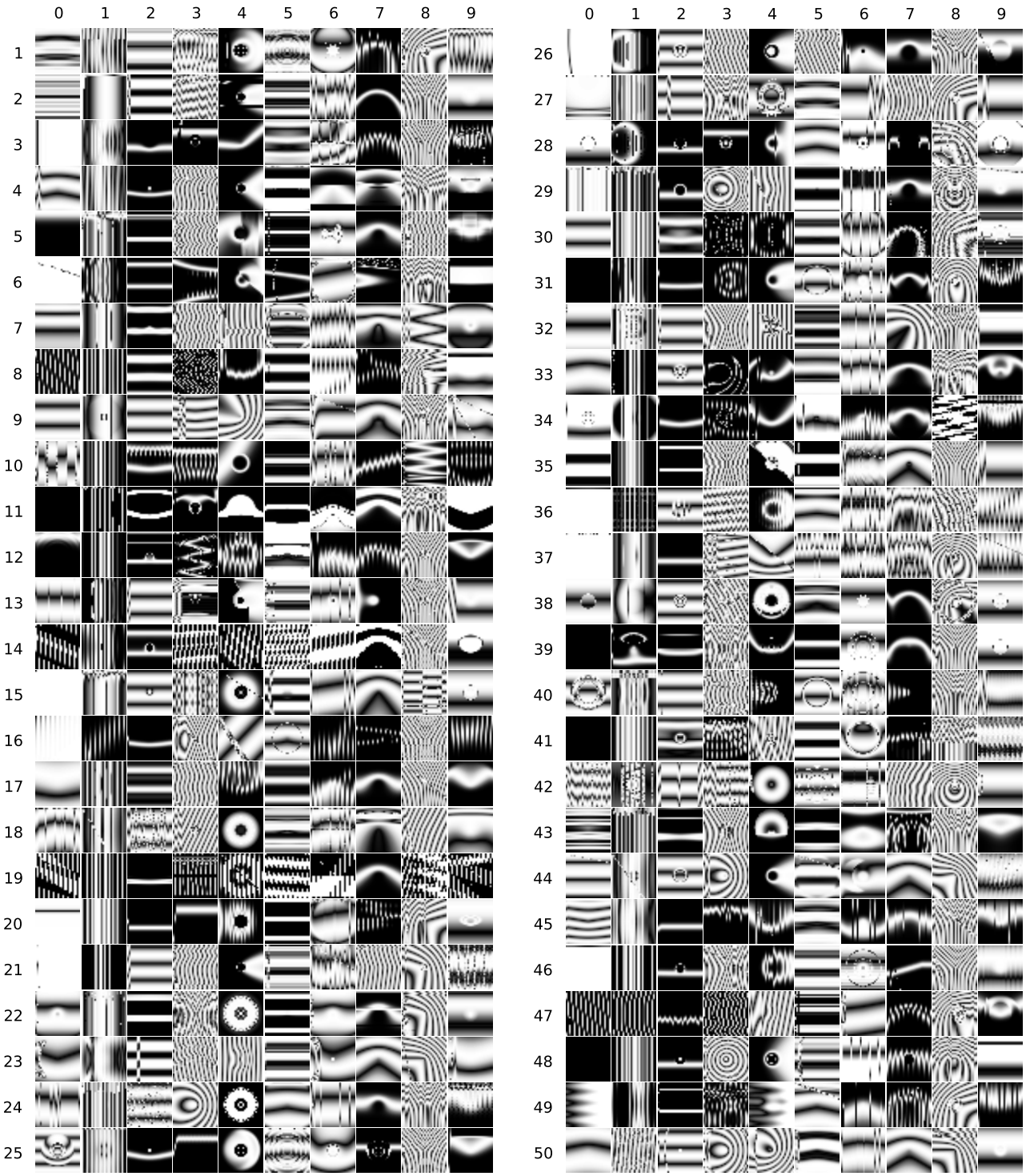

Figure S2. Images from runs 1 through 50. Columns are digits. In each row are the final (best) images evolved for each class during that run.

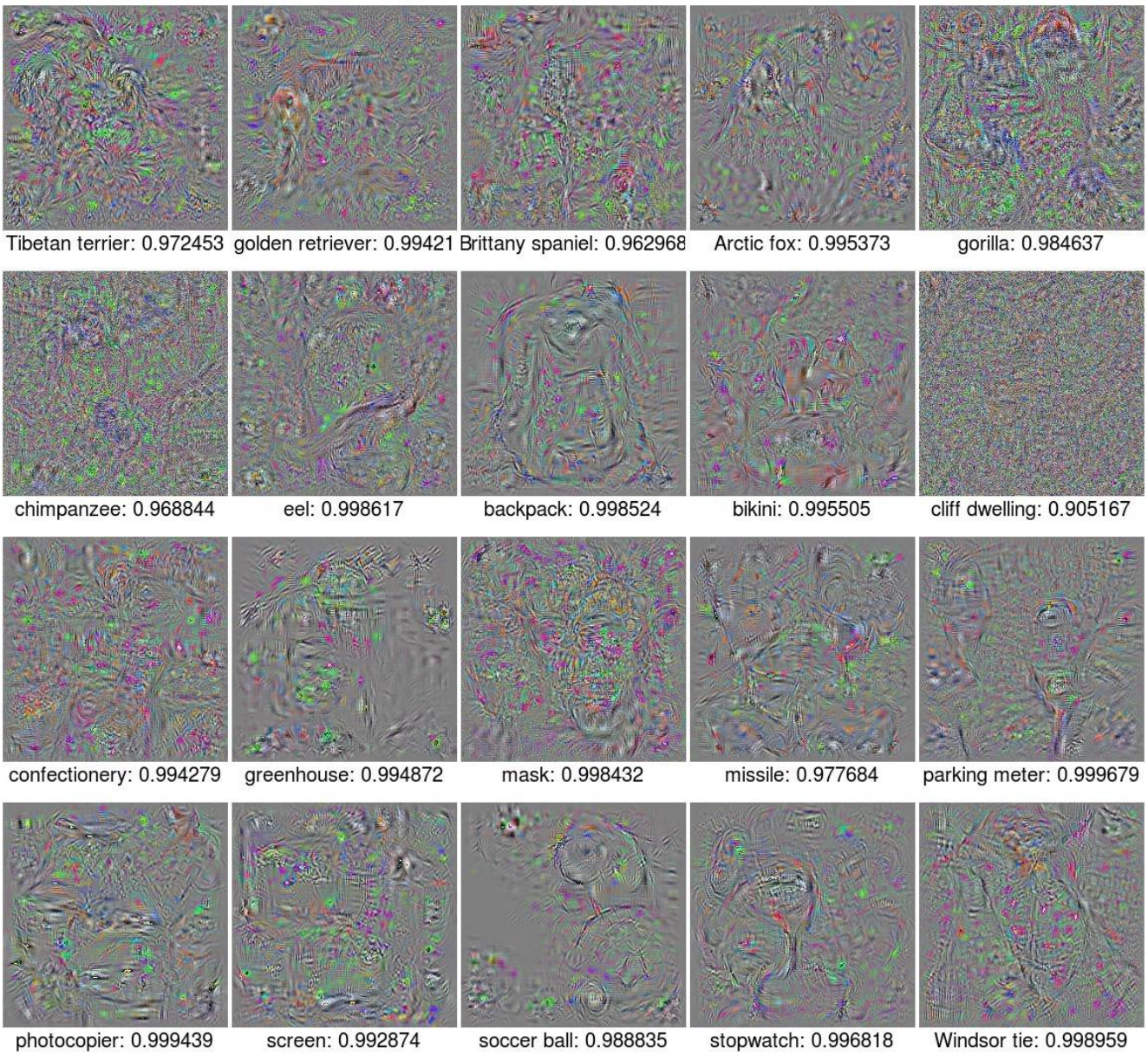

Figure S3. Images found by directly maximizing an objective function consisting of the posterior probability (softmax output) added to a regularization term, here L2-regularization. Optimization begins at the ImageNet mean plus small Gaussian noise to break symmetry. When regularization is added, confidences are generally lower than 99.99% because the objective contains terms other than confidence. Here, the average is 98.591%. For clarity, images are shown with the mean subtracted.

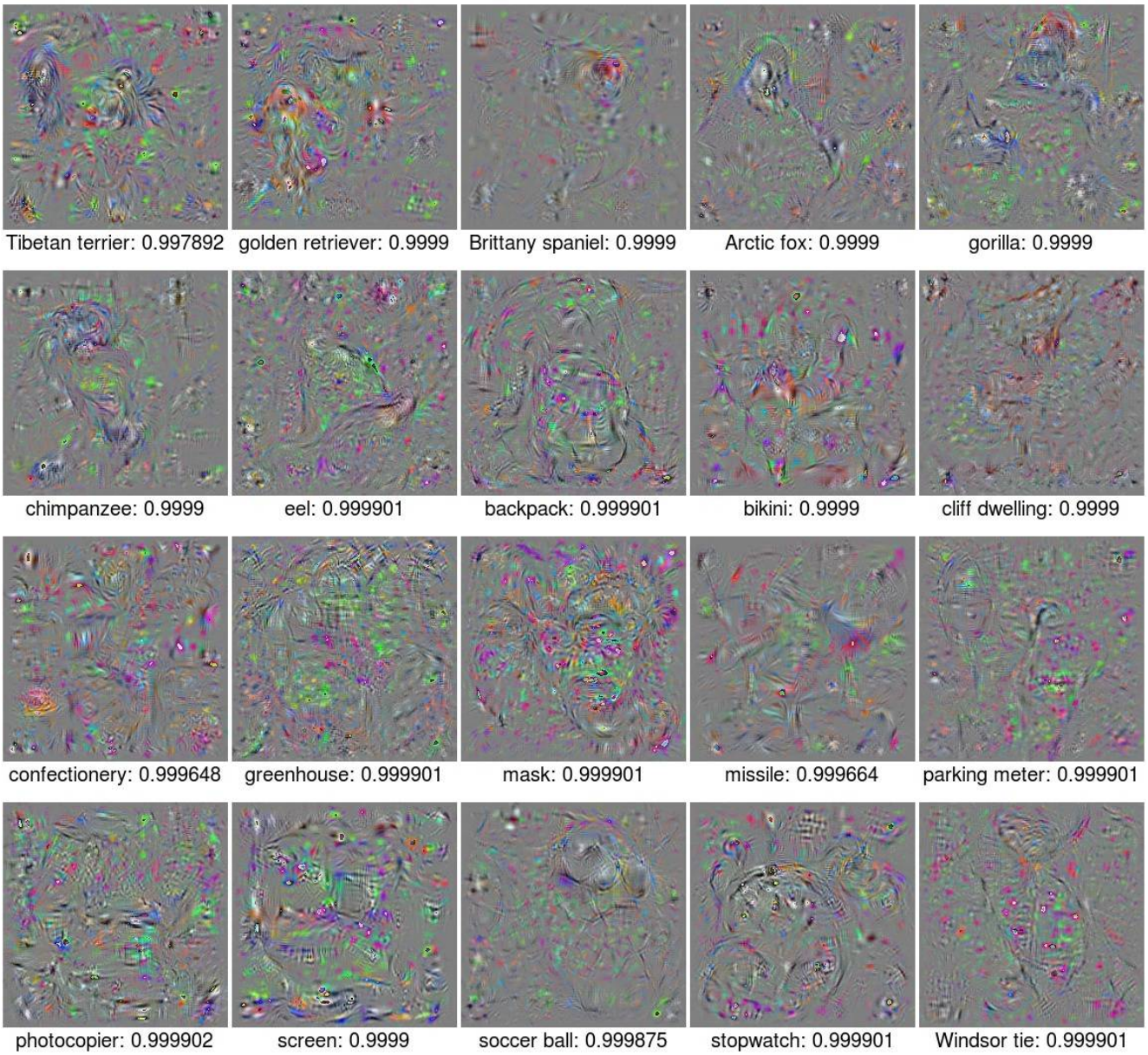

Figure S4. As in Fig. S3, but with blurring and pseudo-L1-regularization, which is accomplished by setting the pixels with lowest norm to zero throughout the optimization.

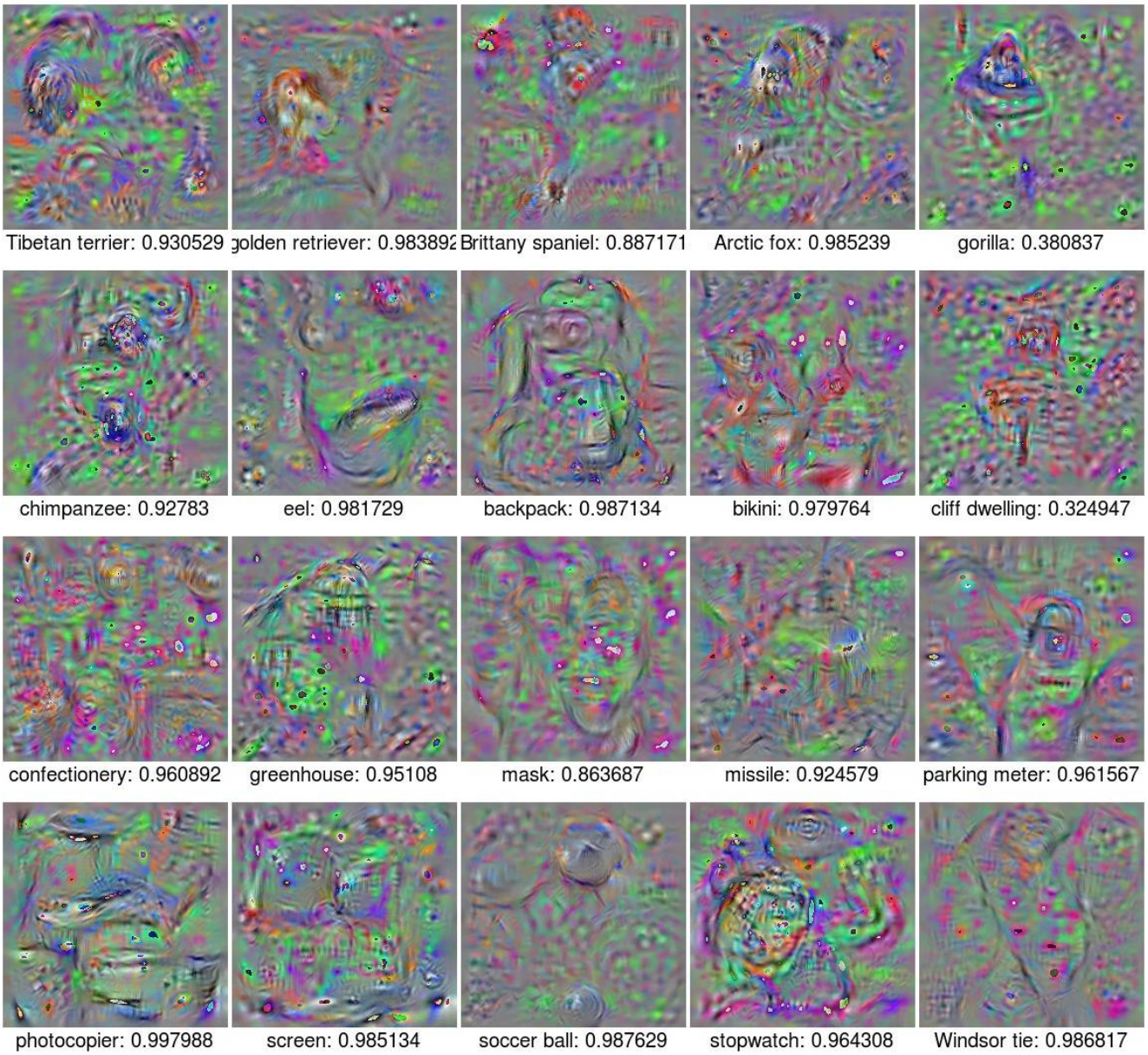

Figure S5. As in Fig. S3, but with slightly more aggressive blurring than in Fig. S4.

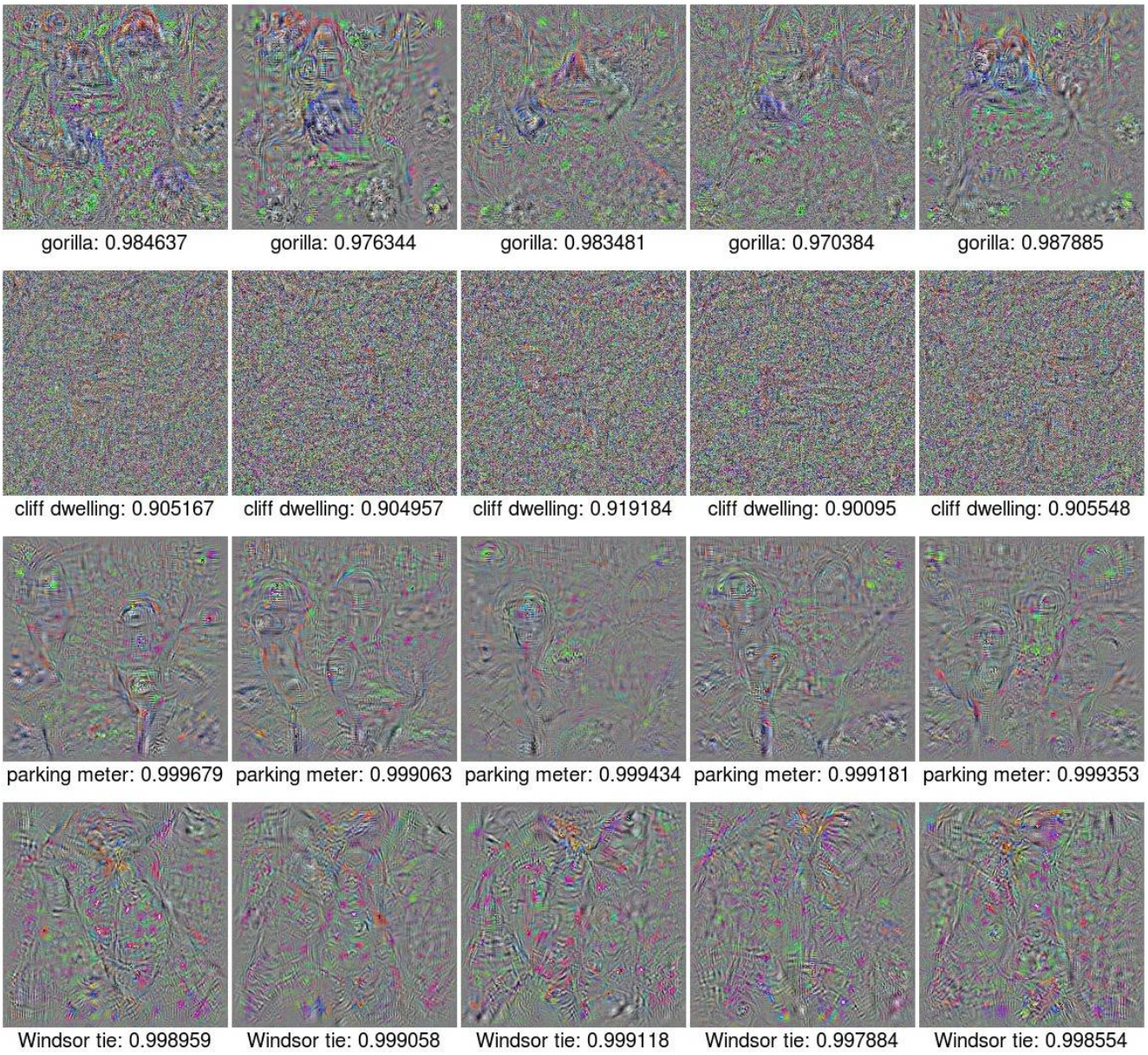

Figure S6. Multiple images produced for each class in the manner of Fig. S3. Each column shows the result of a different local optimum, which was reached by starting at the ImageNet mean and adding different draws of small Gaussian noise.

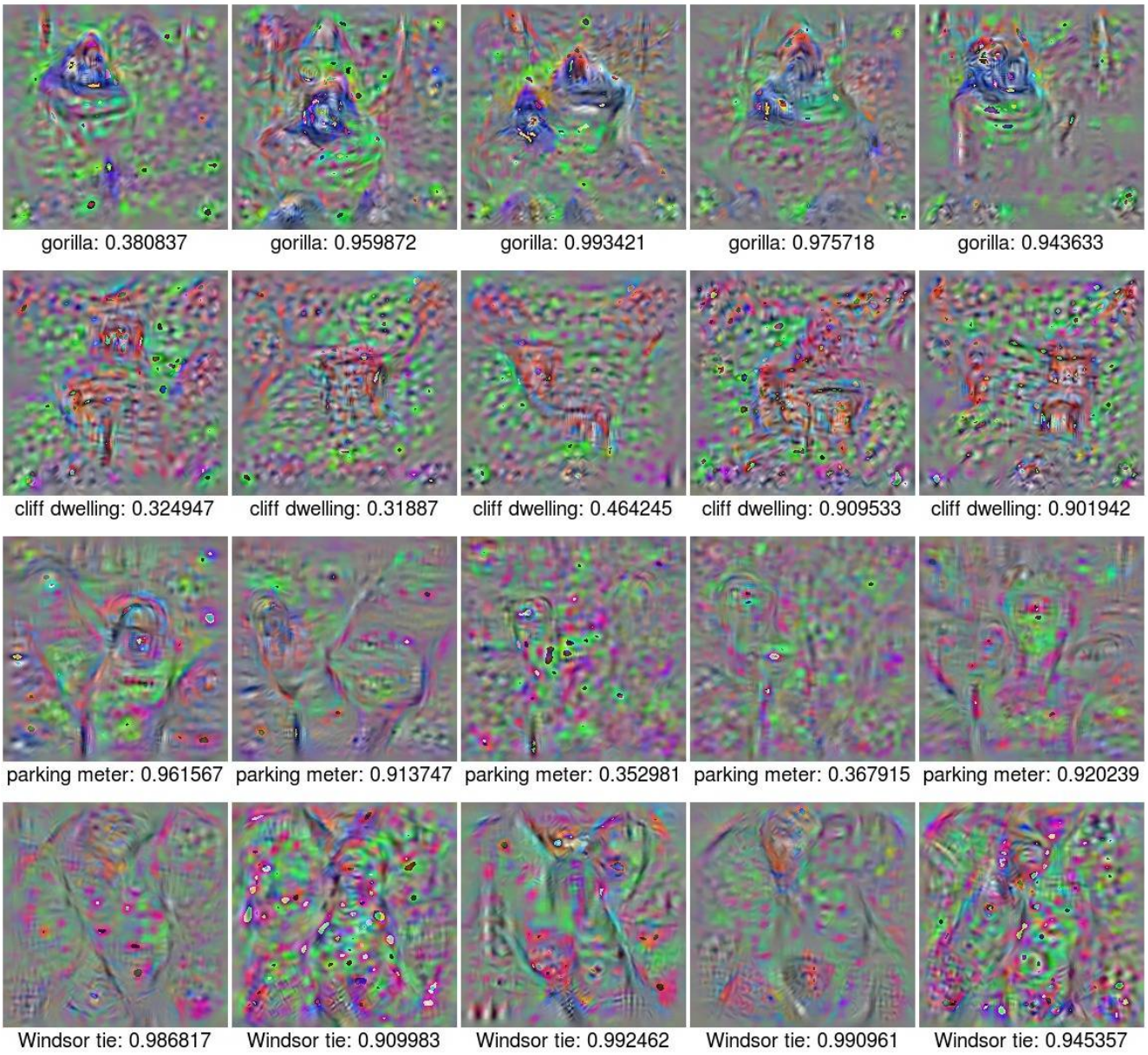

Figure S7. Multiple images produced for each class in the manner of Fig. S5. Each column shows the result of a different local optimum, which was reached by starting at the ImageNet mean and adding different draws of small Gaussian noise.