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

A. Images that fool one DNN generalize to fool other DNNs

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 MNIST and ImageNet DNNs." Here we show the details of this experiment and its results.

A.1. Generalization across DNNs with the same architecture

We performed this test with two MNIST [3] DNNs (MNIST_A and MNIST_B) and two ImageNet [2] DNNs (ImageNet_A and ImageNet_B), where A and B differ only in their random initializations, but have the same architecture. 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.

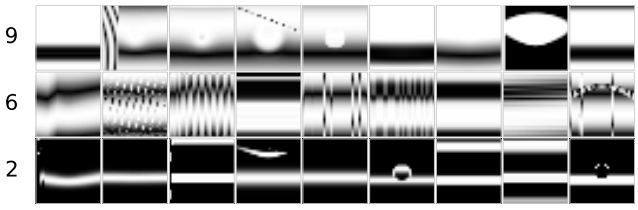

Figure S1. CPPN-encoded, evolved images which are given $\geq$ 99% confidence scores by both DNN_A and DNN_B to represent digits 9, 6, and 2. Each column represents an image produced by an independent run of evolution, yet evolution converges on a similar design, which fools not just the DNN it evolved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

978-1-4673-6964-0/15/$31.00 ©2015 IEEE

A.2. Generalization across DNNs that have different architectures

Here we test whether images that fool a DNN with one architecture also fool another DNN with a different architecture. We performed this test with two well-known ImageNet DNN architectures: AlexNet [2] and GoogLeNet [5], both of which are provided by Caffe [1] and trained on the same ILSVRC 2012 dataset [4]. GoogLeNet has a top-1 error rate of 31.3%.

1000 images were produced with each ImageNet DNN. We found that 20.7% of images evolved for GoogLeNet are also given the same top-1 label by AlexNet (and 17.3% vice versa). Thus, many fooling examples are not fit precisely to a particular network, but generalize across different DNN architectures.

B. Does using an ensemble of networks instead of just one prevent fooling?

We also tested whether requiring an image to fool an ensemble of multiple networks makes it impossible to produce fooling images. We tested an extreme case where each network in the ensemble has a different architecture. Specifically, we tested with an ensemble of 3 different DNN architectures: CaffeNet, AlexNet and GoogLeNet. CaffeNet [1] performs similarly to AlexNet [2], but has a slightly different architecture. The final confidence score given to an image is calculated as the mean of the three scores given by these three different DNNs. After only 4000 generations, evolution was still able to produce fooling images for 231 of the 1000 classes with $\geq 90\%$ confidence. Moreover, the median is also high at 65.2% and the max is 100%.

C. 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 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 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.

C.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 $\sim$25% of the training set images.

C.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 $\sim$7-fold increase over the $\sim$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

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.

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 $\sim$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 1001$^{\text{st}}$ 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).

D. 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. S4).

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.

E. 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. S5, S6, and S7.

Fig. S5 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. S5 shows $\gamma = 0.01$.

Fig. S6 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. S7 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. S8 and Fig. S9 show the variety of images that may be produced by taking different random draws of this initial noise.

F. Confidence scores of real ImageNet images

The optimization methods presented can generate unrecognizable images that are given high confidence scores. However, to find out if these high scores for fooling images are similar to the confidence scores given by DNNs for the natural images they were trained to classify, we evaluate the entire ImageNet validation set with the ImageNet DNN [2]. Across 50,000 validation images, the median confidence score is 60.3%. Across the cases when images are classified correctly (i.e., the top-1 label matches the ground truth label), the DNN gives a median confidence score of 86.7%. On the contrary, when the top-1 prediction label does not match the ground truth, the images are given only 33.7% median confidence. Thus, the median confidence score of 88.11% of synthetic images that match ImageNet is comparable to that of real images.

G. Can the fooling images be considered art?

To test the hypothesis that the CPPN fooling images could actually be considered art, we submitted a selection of them to a selective art contest: the "University of Wyoming 40th Annual Juried Student Exhibition", which only accepted 35.5% of the submissions. Not only were the images accepted, but they were also amongst the 21.3% of submissions to be given an award. The work was then displayed at the University of Wyoming Art Museum (Fig. S2, S3). The submitted image is available at http://evolvingai.org/fooling.

Figure S2. A selection of fooling images were accepted as art in a selective art competition. They were then displayed alongside human-made art at a museum.

Figure S3. Museum visitors view a montage of CPPN-encoded fooling images.

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, 2, 4

[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

[4] 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. *arXiv preprint arXiv:1409.0575*, 2014. 2

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 2

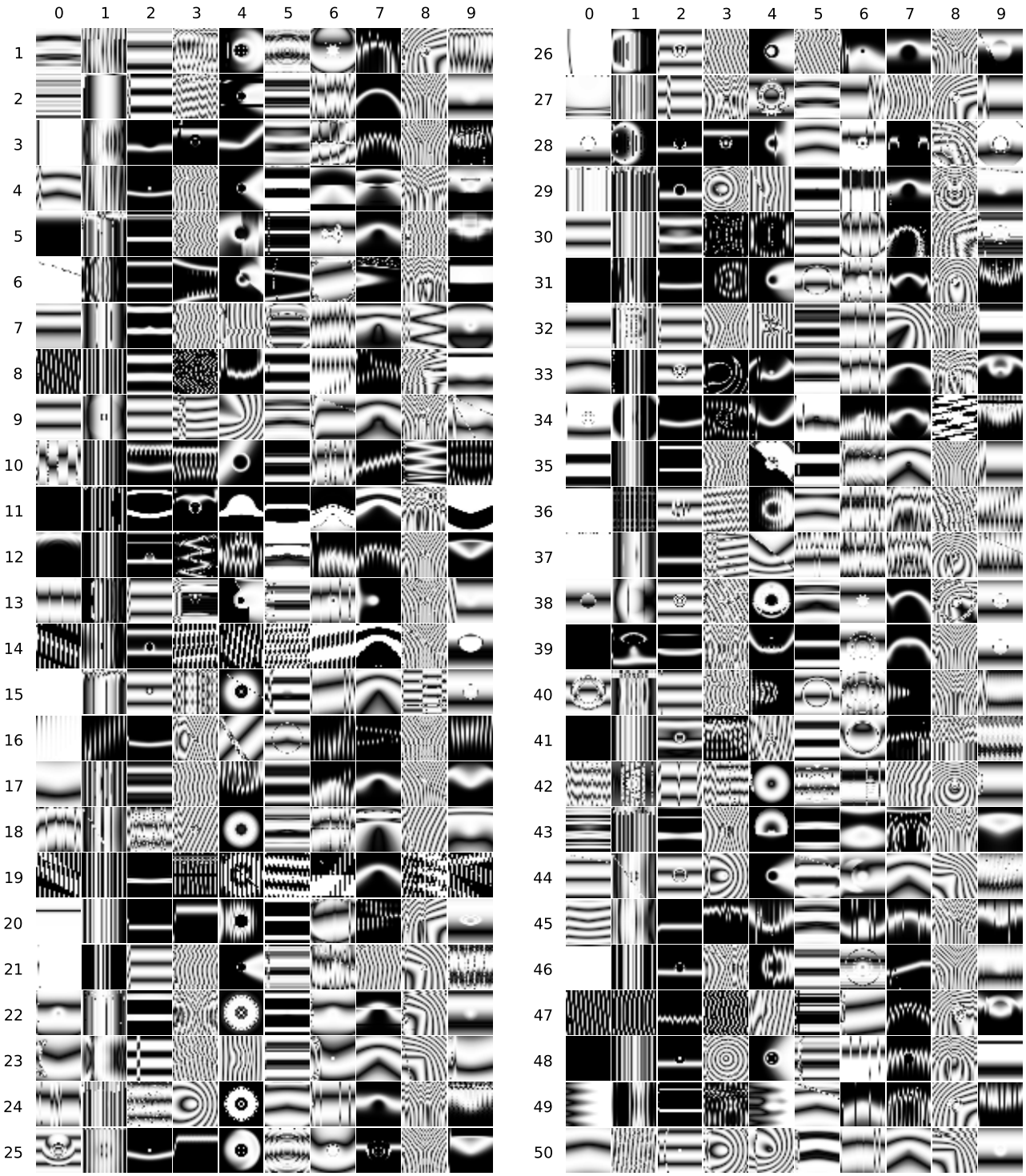

Figure S4. 50 independent runs of evolution produced images that an MNIST DNN believes with 99.99% to be handwritten digits. Columns are digits. In each row are the final (best) images evolved for each class during that run.

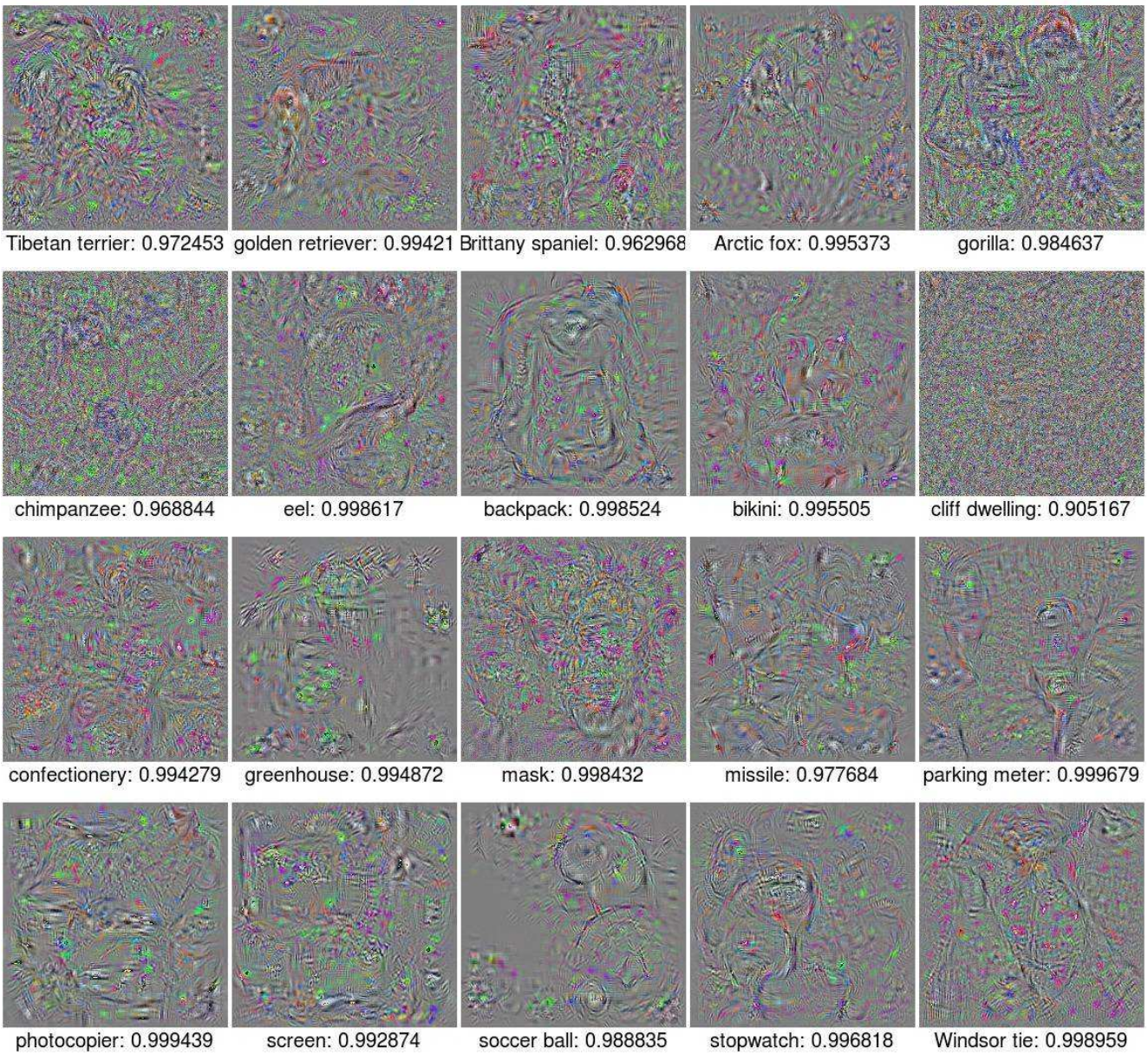

Figure S5. Images found by directly maximizing an objective function consisting of the posterior probability (softmax output) added to a regularization term, here L2-regularization. Optimization begins at the ImageNet mean plus small Gaussian noise to break symmetry. When regularization is added, confidences 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.

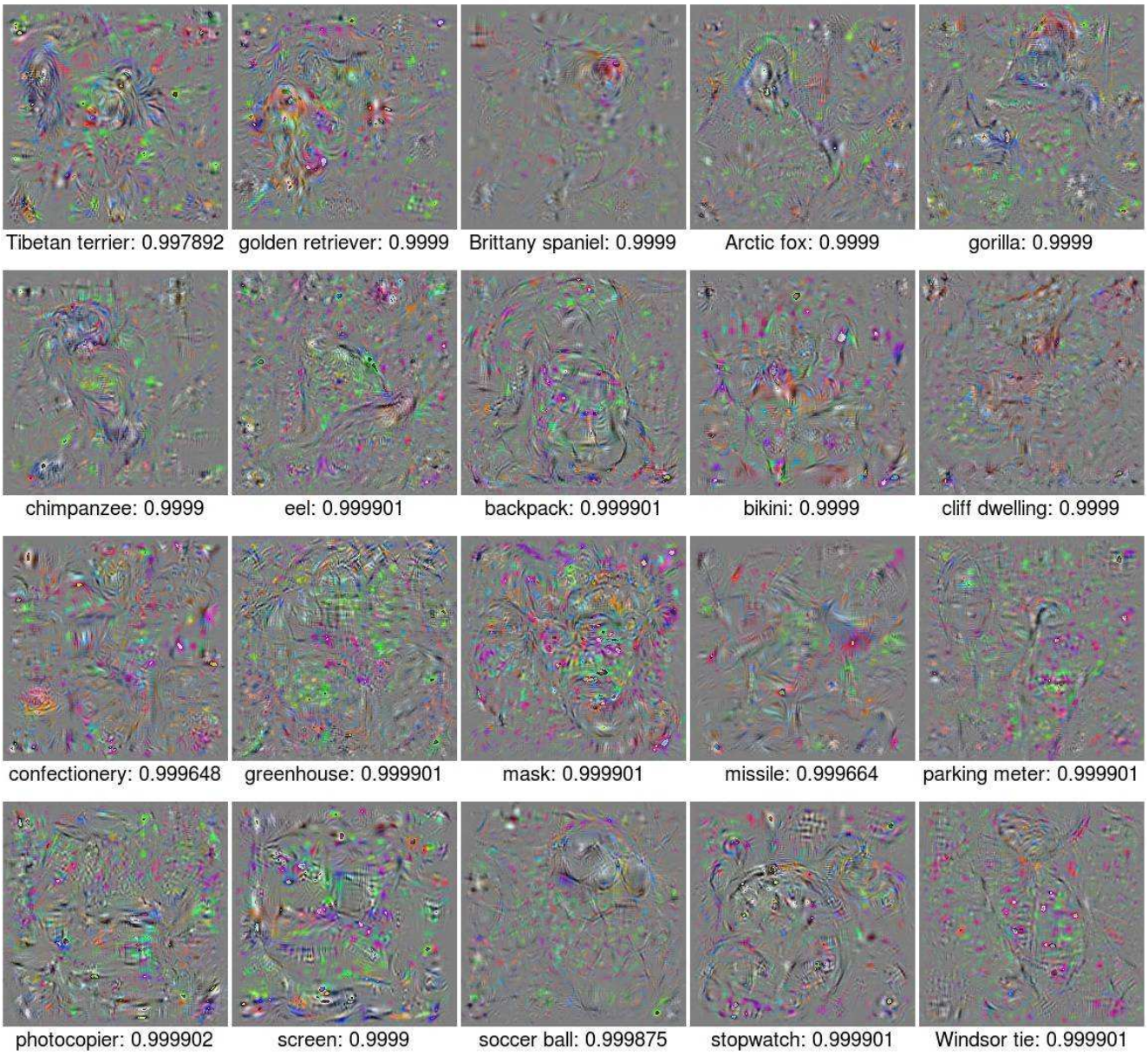

Figure S6. As in Fig. S5, but with blurring and pseudo-L1-regularization, which is accomplished by setting the pixels with lowest norm to zero throughout the optimization.

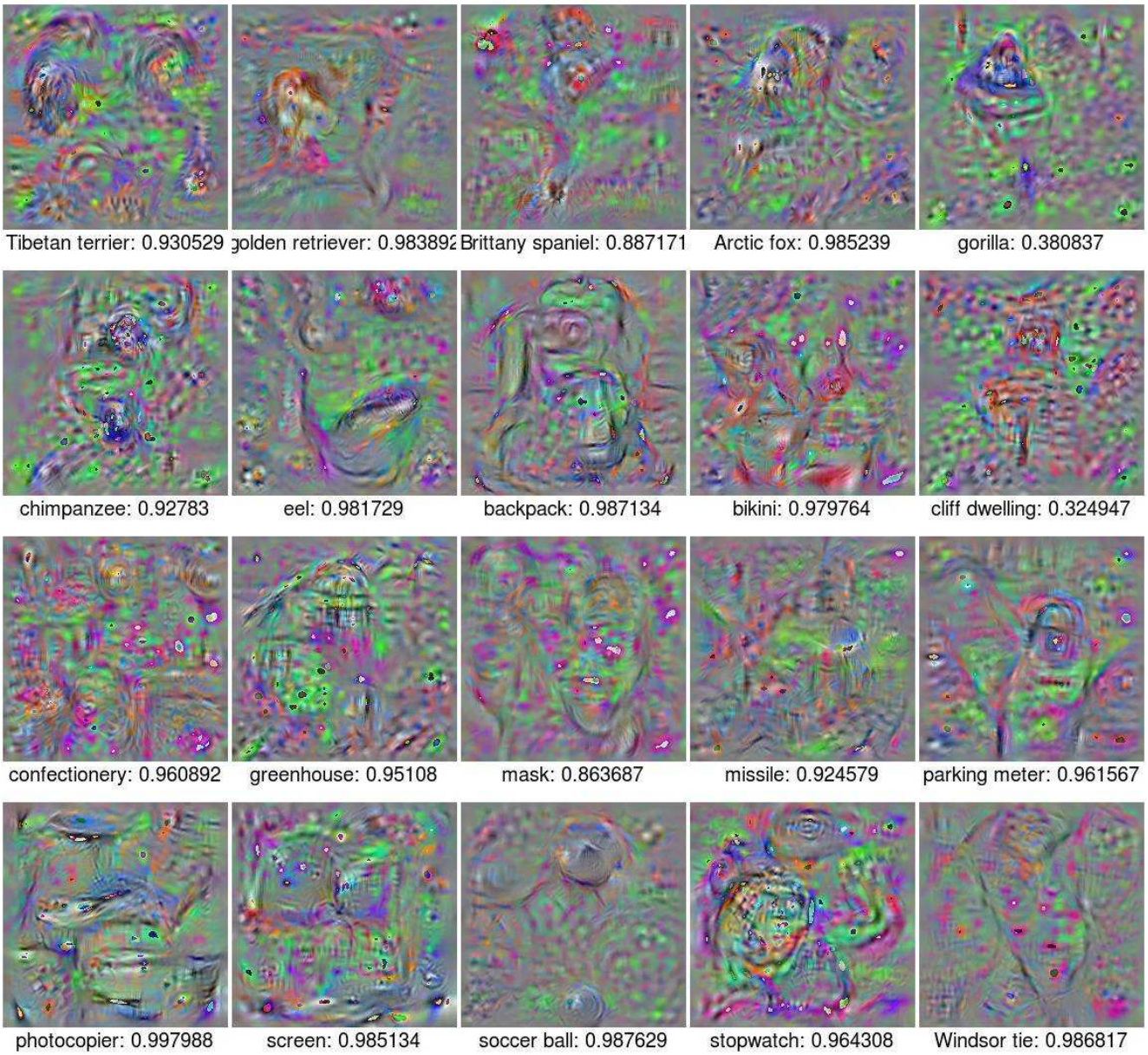

Figure S7. As in Fig. S5, but with slightly more aggressive blurring than in Fig. S6.

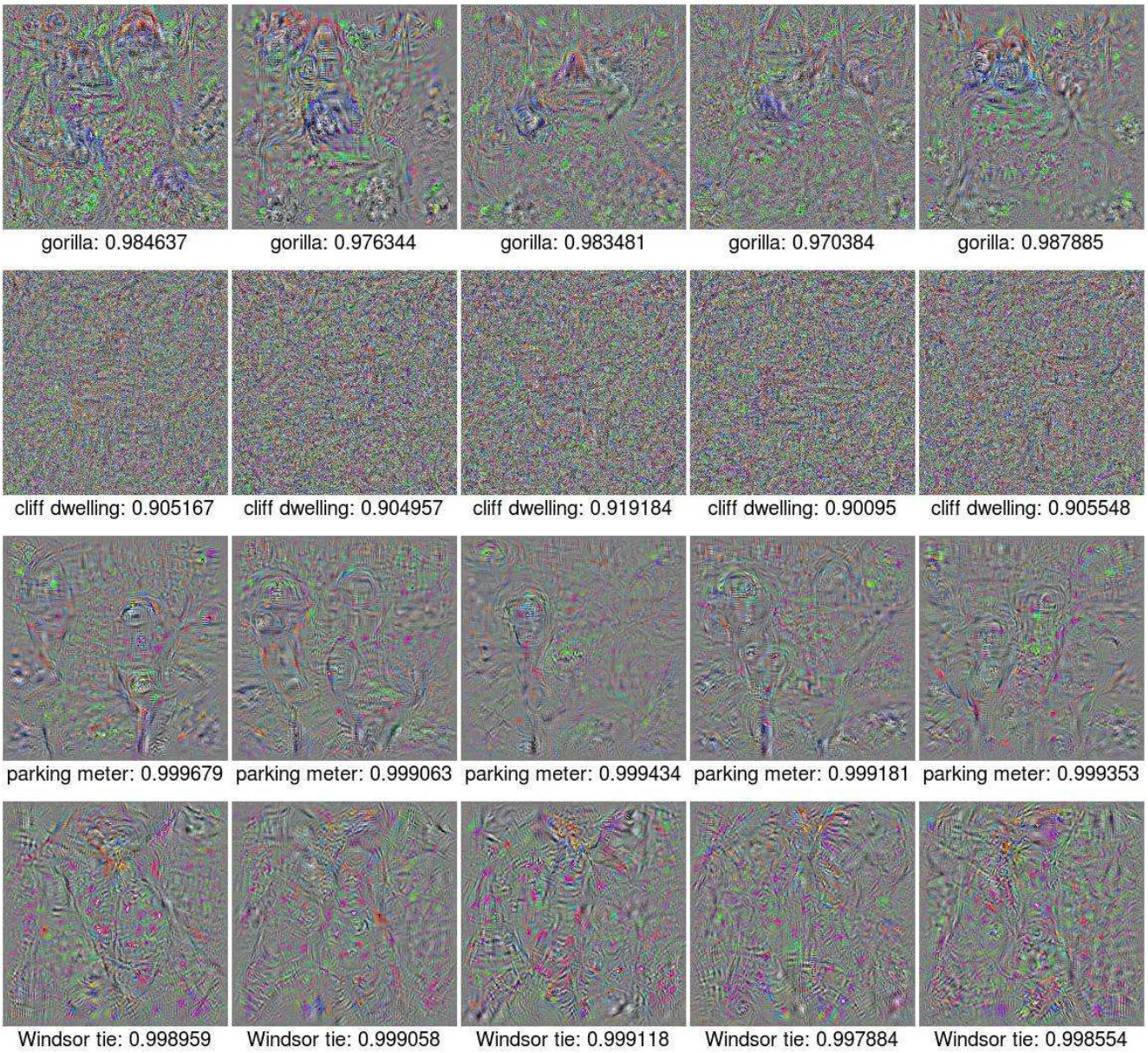

Figure S8. Multiple images produced for each class in the manner of Fig. S5. Each column shows the result of a different local optimum, which was reached by starting at the ImageNet mean and adding different draws of small Gaussian noise.

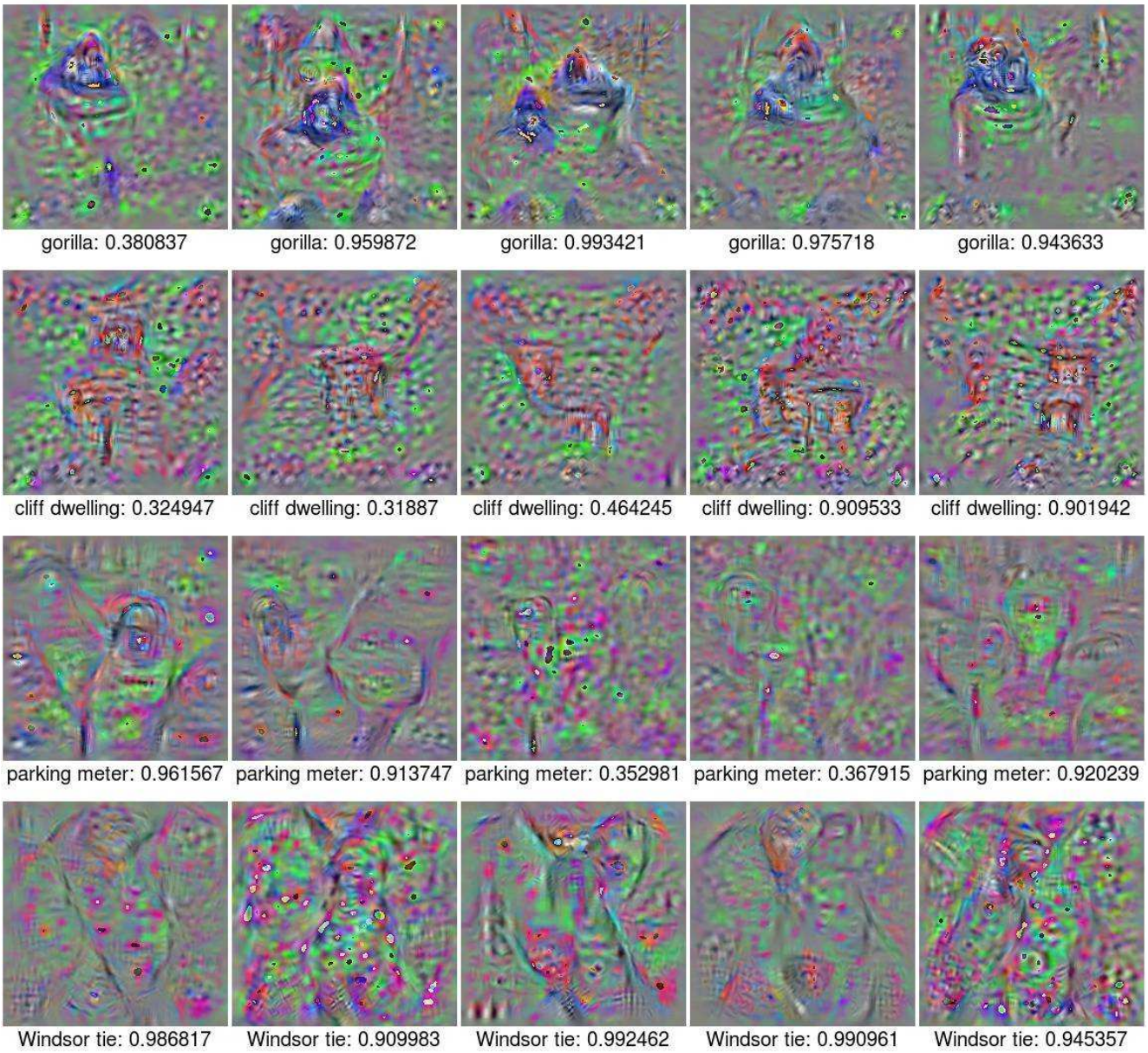

Figure S9. Multiple images produced for each class in the manner of Fig. S7. Each column shows the result of a different local optimum, which was reached by starting at the ImageNet mean and adding different draws of small Gaussian noise.