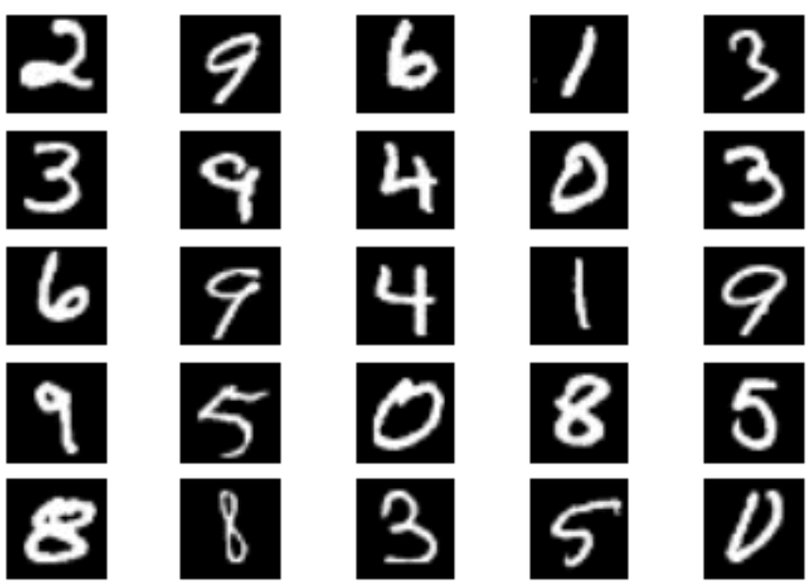
Fooling Neural Networks

Linguang Zhang
Feb-4-2015

Preparation

- Task: image classification.
- Datasets: MNIST, ImageNet.
- training and testing data.



Preparation

- Logistic regression:
- Good for 0/1 classification. e.g. spam filtering

$$h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)}$$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right]$$

Preparation

- Multi-class classification? N categories?
- Softmax regression
- Weight Decay (regularization)

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1|x^{(i)};\theta) \\ p(y^{(i)} = 2|x^{(i)};\theta) \\ \vdots \\ p(y^{(i)} = k|x^{(i)};\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right] + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}\theta_{ij}^2$$

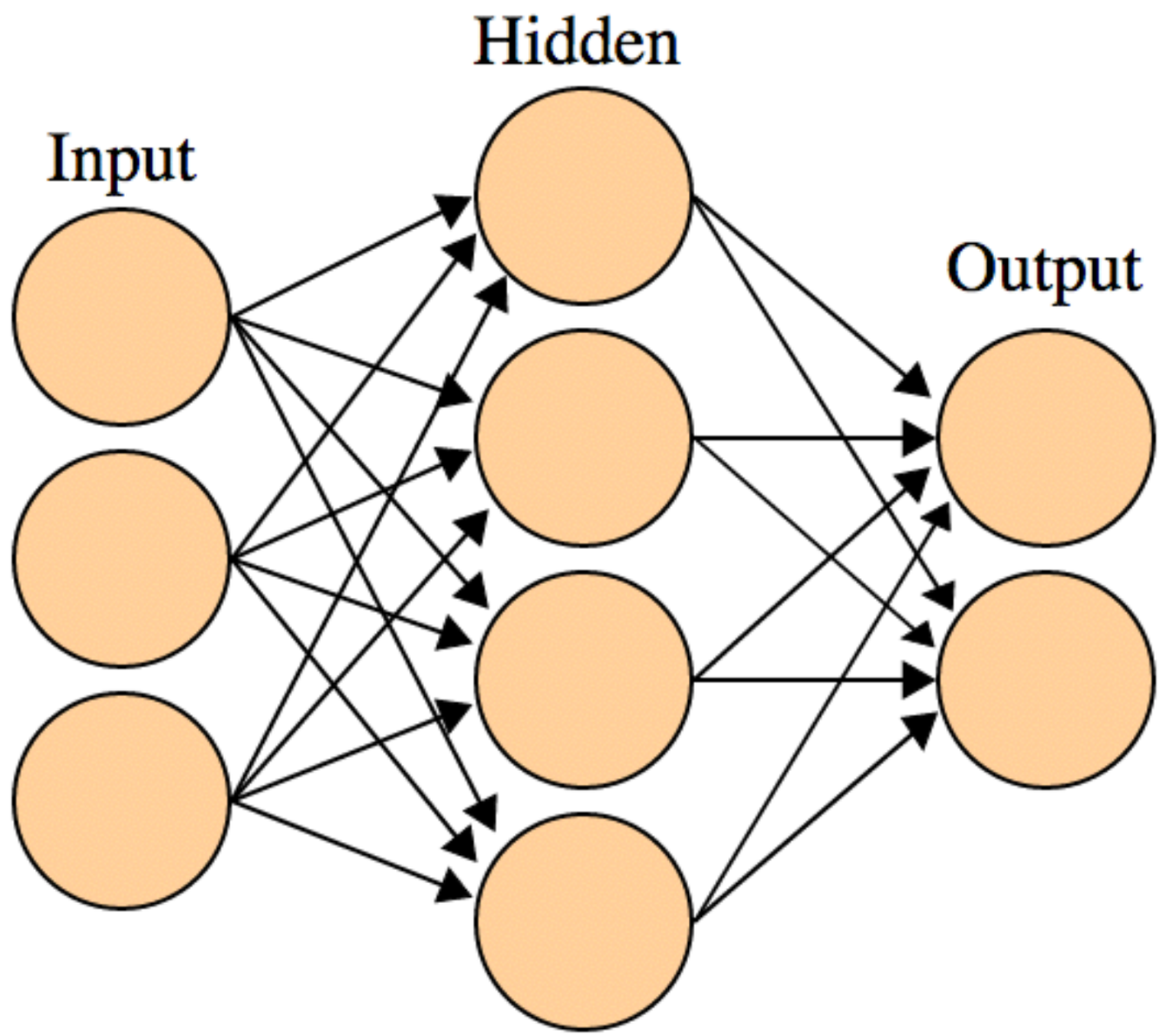
Preparation

- Autoencoder
- What is autoencoder?

  Input = decoder(encoder(input))

- Why is it useful? Dimension reduction.
- Training
  - Feed-forward and obtain output $\hat{x}$ at the output layer
  - Compute dist($\hat{x}$, x).
  - Update weights through backpropagation.

Basic Neural Network



Intriguing Properties of Neural Networks

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013).

Activation $\phi(x)$

- Activation of a hidden unit is a meaningful feature.

using the natural basis of the i-th hidden unit:

$$x' = \arg\max_{x \in \text{image set}} \langle \phi(x), e_i \rangle$$

randomly choose a vector:

$$x' = \arg\max_{x \in \text{image set}} \langle \phi(x), v \rangle$$

using the natural basis:

(b) Unit sensitive to upper round stroke, or lower straight stroke.

(d) Unit senstive to diagonal straight stroke.

randomly choose a vector:

(b) Direction sensitive to lower left loop.

(d) Direction sensitive to right, upper round stroke.

Adversarial Examples

- What is adversarial example? We can let the network to misclassify an image by adding a imperceptible (for human) perturbation.

- Why do adversarial examples exist? Deep Neural Networks learn input-output mappings that are discontinuous to a significant extent.

- Interesting observation: the adversarial examples generated for network A can also make network B fail.

Generate Adversarial Examples

Input image: $x \in \mathbb{R}^m$

Classifier: $f : \mathbb{R}^m \longrightarrow \{1 \ldots k\}$

Target label: $l \in \{1 \ldots k\}$

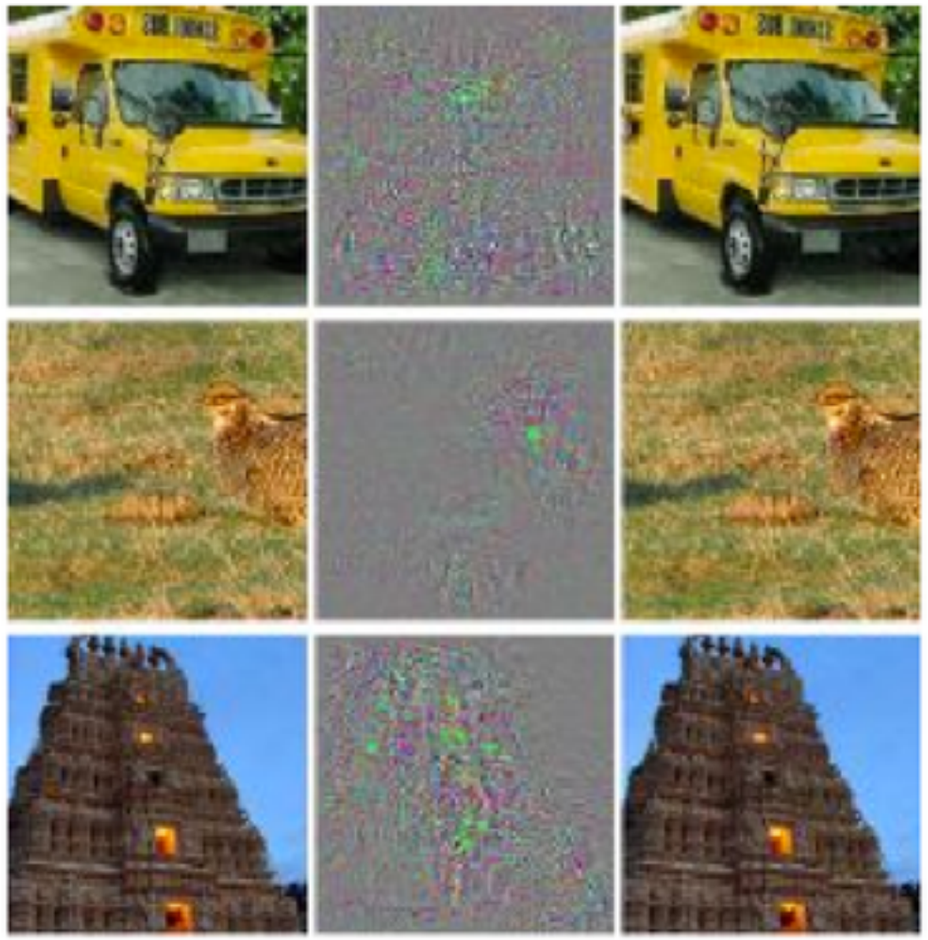
Minimize $\|r\|_2$ subject to:

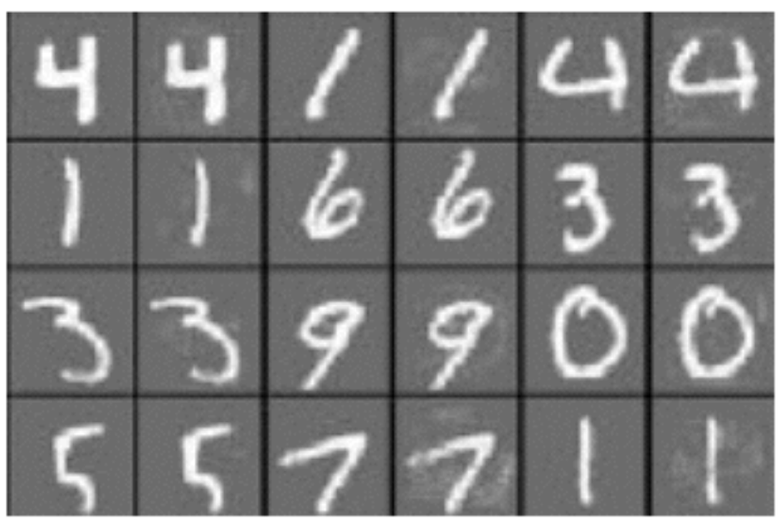
1. $f(x + r) = l$
2. $x + r \in [0, 1]^m$

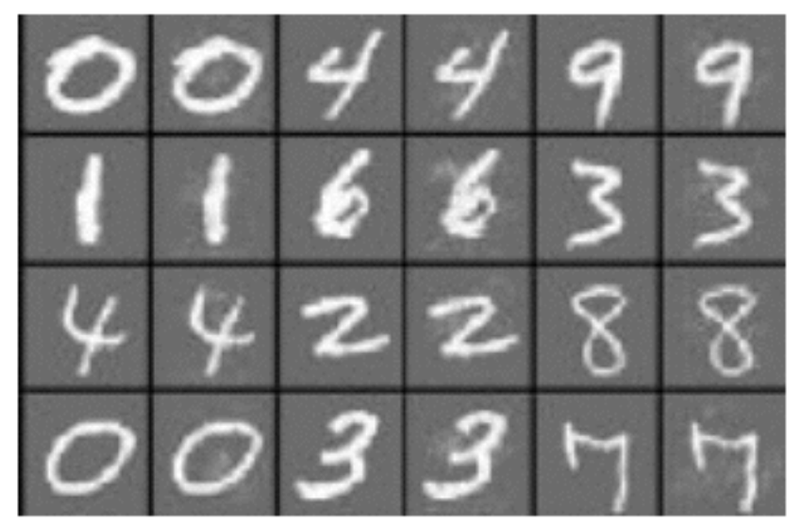
x+r is the closest image to ***x*** classified as ***l*** by ***f***.

When $f(x) \neq l$:

Minimize $c|r| + \text{loss}_f(x + r, l)$ subject to $x + r \in [0, 1]^m$







Intriguing properties

- Properties:
  - Visually hard to distinguish the generated adversarial examples.
  - Cross model generalization. (different hyper-parameters)
  - Cross training-set generalization. (different training set)
- Observation:
  - adversarial examples are universal.
  - back-feeding adversarial examples to training might improve generalization of the model.

Experiment

Cross-model generalization of adversarial examples.

	softmax1	softmax2	softmax3	N100-100-10	N200-200-10	AE400-10	Av. distortion
softmax with $\lambda = 10^{-4}$	100%	11.7%	22.7%	2%	3.9%	2.7%	0.062
softmax with $\lambda = 10^{-2}$	87.1%	100%	35.2%	35.9%	27.3%	9.8%	0.1
softmax with $\lambda = 1$	71.9%	76.2%	100%	48.1%	47%	34.4%	0.14
N100-100-10	28.9%	13.7%	21.1%	100%	6.6%	2%	0.058
N200-200-10	38.2%	14%	23.8%	20.3%	100%	2.7%	0.065
AE400-10	23.4%	16%	24.8%	9.4%	6.6%	100%	0.086
Gaussian noise, stddev=0.1	5.0%	10.1%	18.3%	0%	0%	0.8%	0.1
Gaussian noise, stddev=0.3	15.6%	11.3%	22.7%	5%	4.3%	3.1%	0.3

Experiment

Cross training-set generalization - baseline (no distortion)

Model	Error on P_1	Error on P_2	Error on Test	Min Av. Distortion
M_1: 100-100-10 trained on P_1	0%	2.4%	2%	0.062
M_1': 123-456-10 trained on P_1	0%	2.5%	2.1%	0.059
M_2: 100-100-10 trained on P_2	2.3%	0%	2.1%	0.058

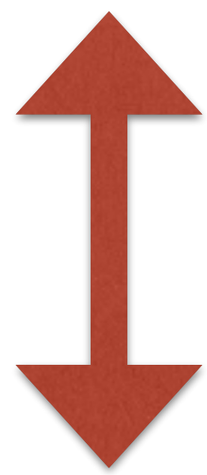
Cross training-set generalization error rate

	M_1	M_1'	M_2
Distorted for M_1 (av. stddev=0.062)	100%	26.2%	5.9%
Distorted for M_1' (av. stddev=0.059)	6.25%	100%	5.1%
Distorted for M_2 (av. stddev=0.058)	8.2%	8.2%	100%
Gaussian noise with stddev=0.06	2.2%	2.6%	2.4%
Distorted for M_1 amplified to stddev=0.1	100%	98%	43%
Distorted for M_1' amplified to stddev=0.1	96%	100%	22%
Distorted for M_2 amplified to stddev=0.1	27%	50%	100%
Gaussian noise with stddev=0.1	2.6%	2.8%	2.7%

magnify distortion

The Opposite Direction

Imperceptible adversarial examples that cause misclassification.

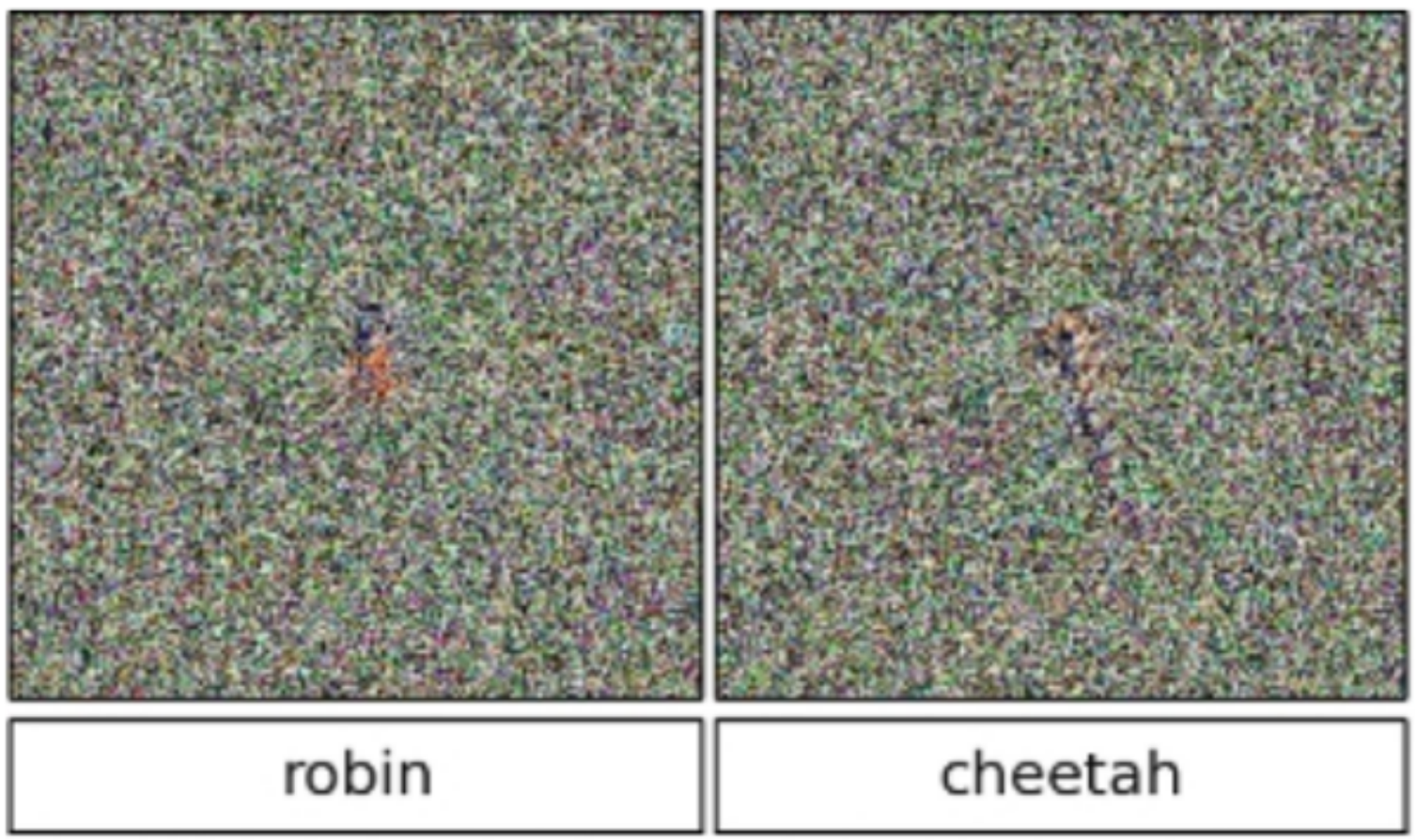


Unrecognizable images that make DNN believe

Nguyen, Anh, Jason Yosinski, and Jeff Clune. "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images." arXiv preprint arXiv:1412.1897 (2014).

Fooling Examples

Problem statement: producing images that are completely unrecognizable to humans, but that state-of-the-art Deep Neural Networks believe to be recognizable objects with high confidence (99%).



DNN Models

- ImageNet: AlexNet. (Caffe version)
  - 42.6% error rate. Original error rate is 40.7%.
- MNIST: LeNet (Caffe version)
  - 0.94% error rate. Original error rate is 0.8%.

Generating Images with Evolution (one class)

- Evolutionary Algorithms (EAs) are inspired by Darwinian evolution.
- Contains a population of organisms (images).
- Organisms will be randomly perturbed and selected based on *fitness function*.
- Fitness function: in our case, is the highest prediction value a DNN believes that the image belongs to a class.

Generating Images with Evolution (multi-class)

- Algorithm: Multi-dimensional archive of phenotypic elites MAP-Elites.

- Procedures:

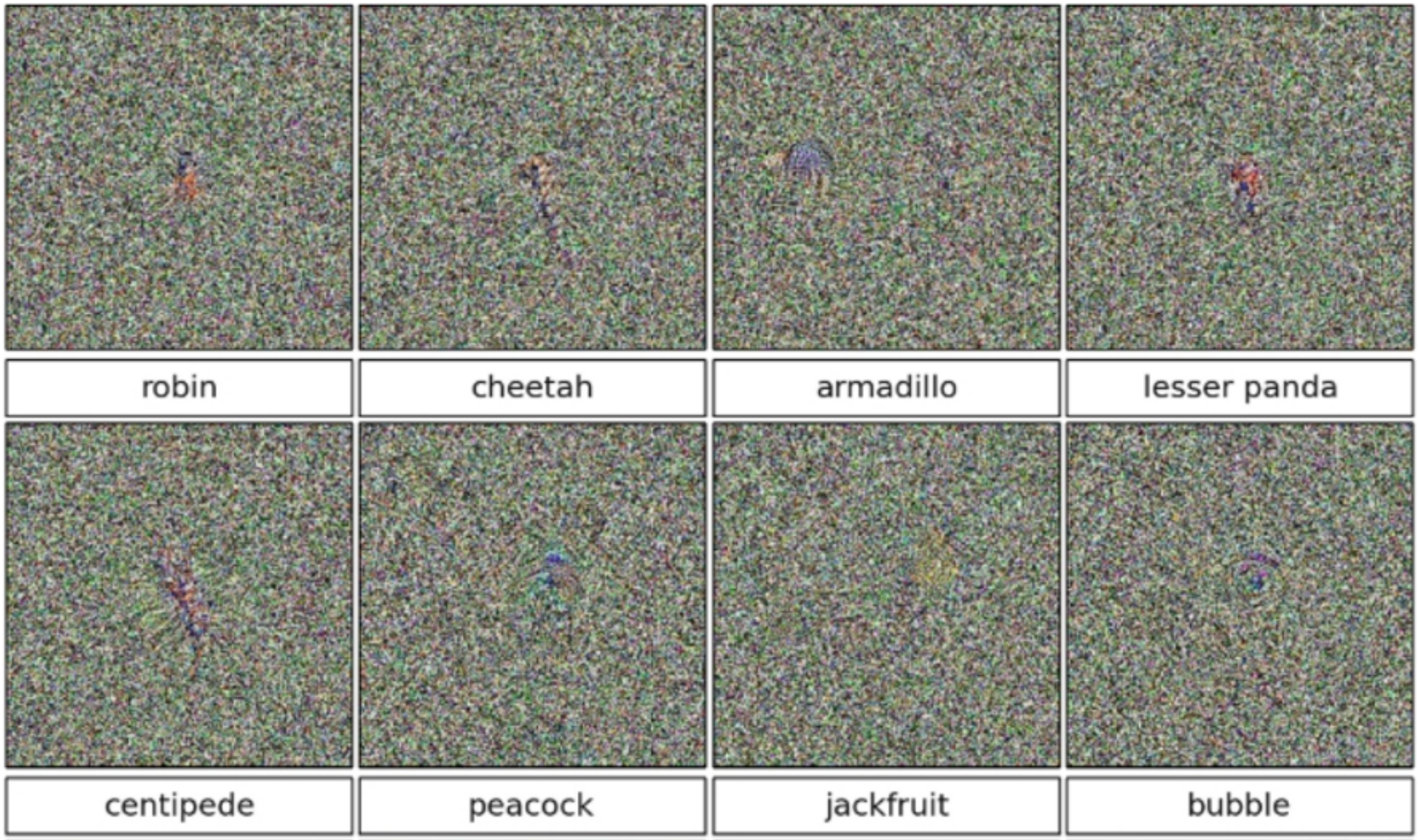
  - Randomly choose an organism, mutate it randomly.

  - Show the mutated organism to the DNN. If the prediction score is higher than the current highest score of ***ANY*** class, make the organism as the champion of that class.

Encoding an Image

- Direct encoding:
  - For MNIST: 28 x 28 pixels.
  - For ImageNet: 256 x 256 pixels, each pixel has 3 channels (H, S, V).
  - Values are independently mutated.
    - 10% chance of being chosen. The chance drops by half every 1000 generations.
    - mutate via the polynomial mutation operator.

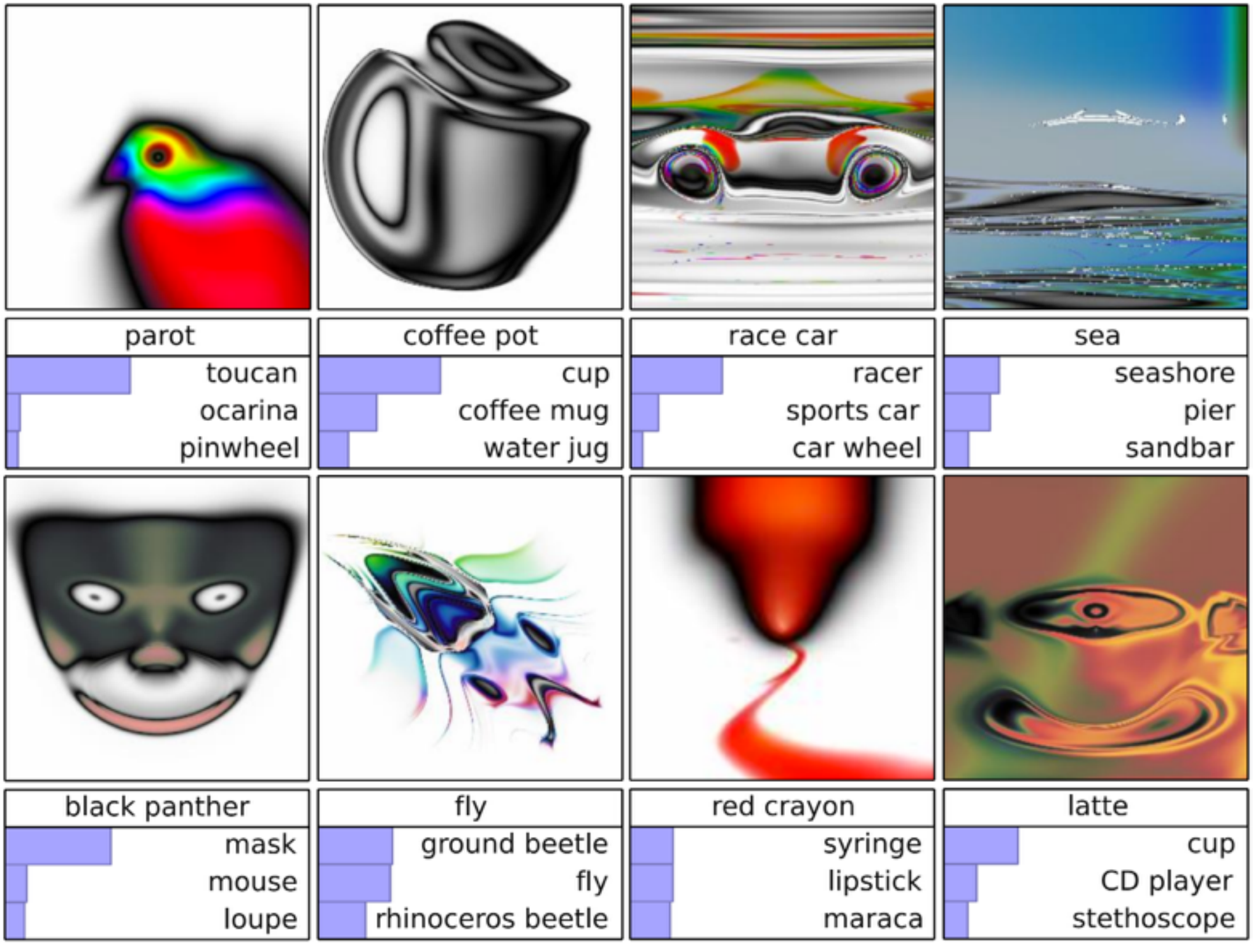
Directly Encoded Images



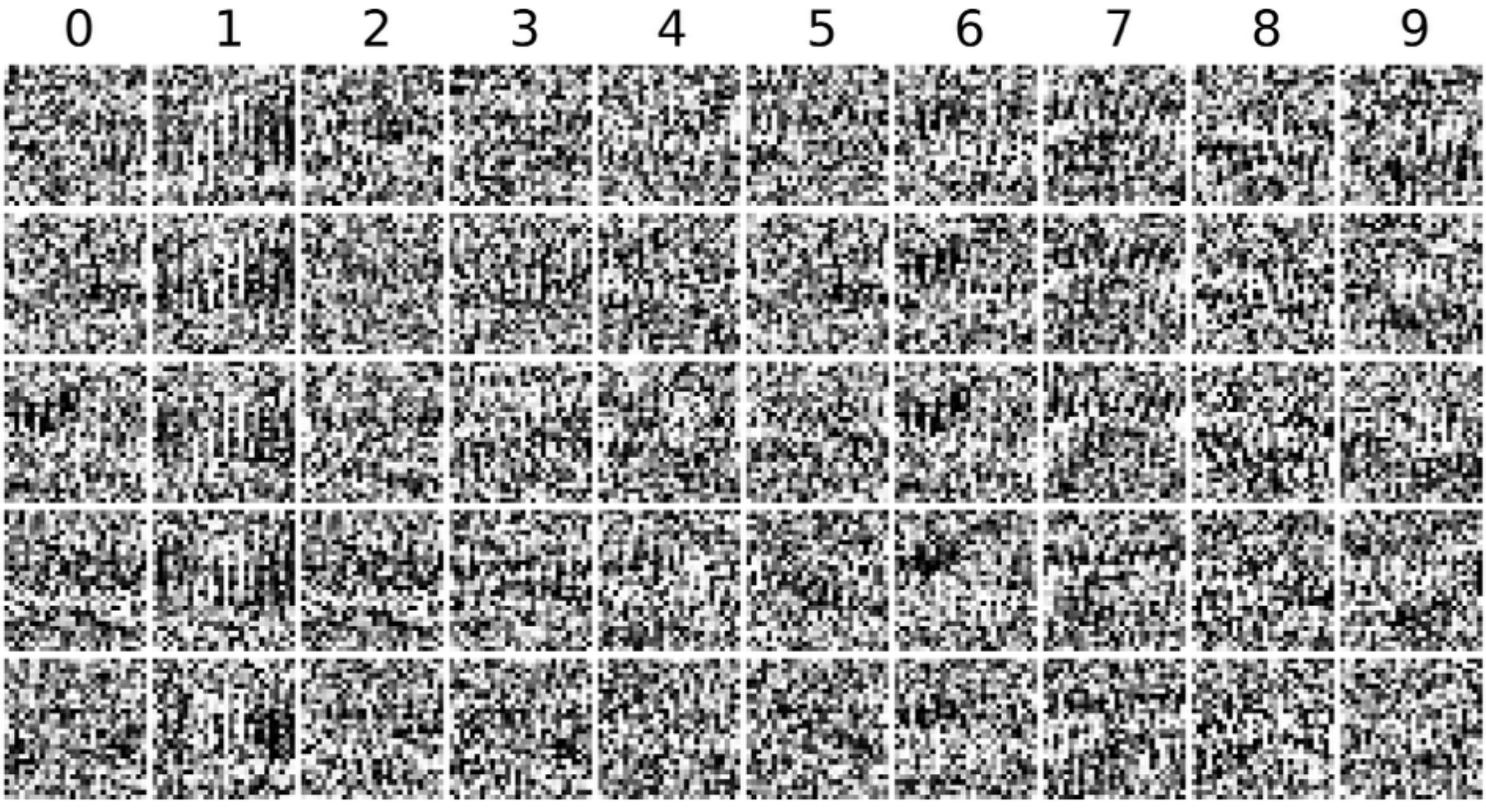
Encoding an Image

- Indirect encoding:
  - very likely to produce regular images with meaningful patterns.
  - both humans and DNNs can recognize.
  - Compositional pattern-producing network (CPPN).

CPPN-encoded Images

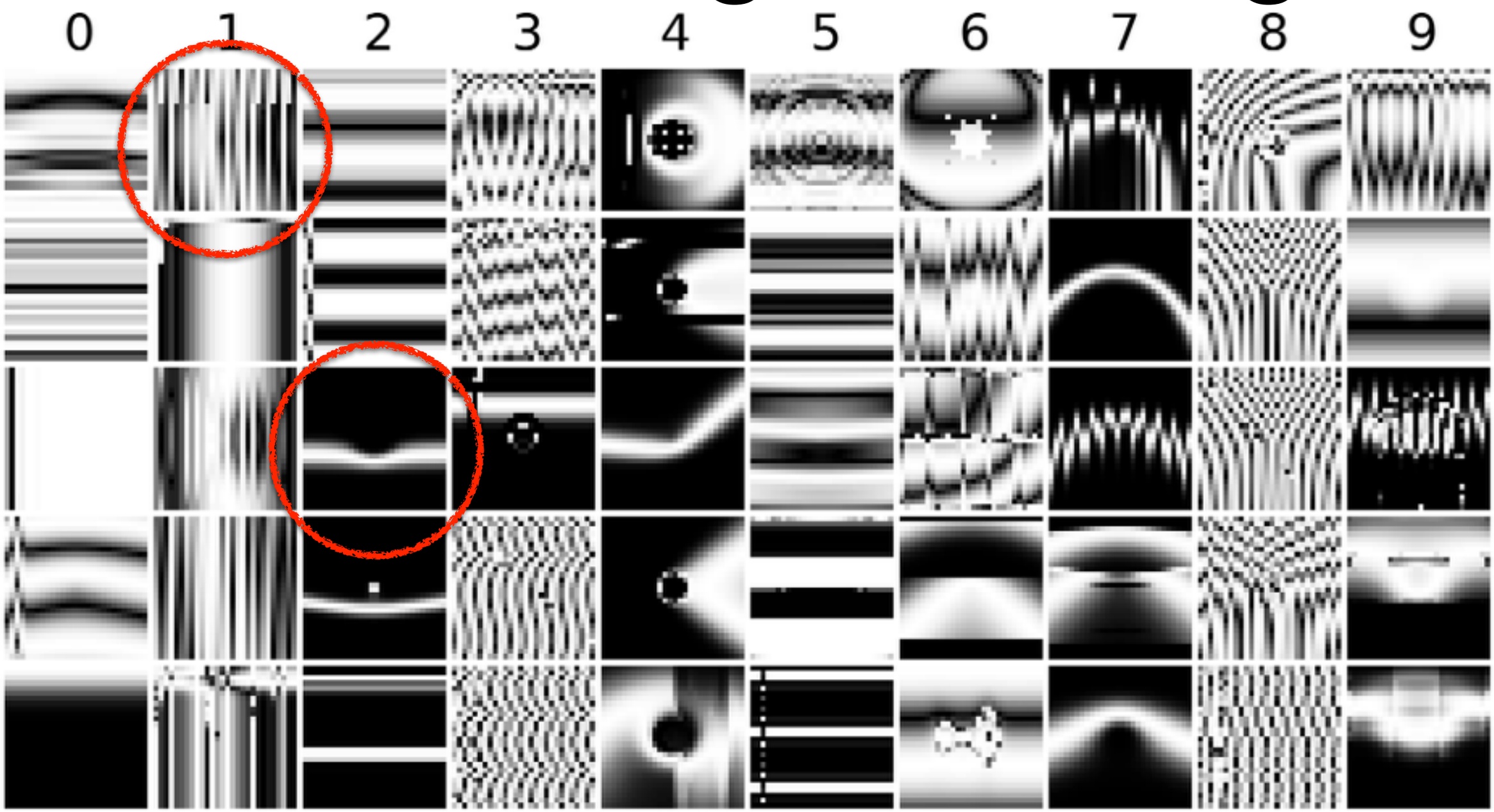


MNIST - Irregular Images



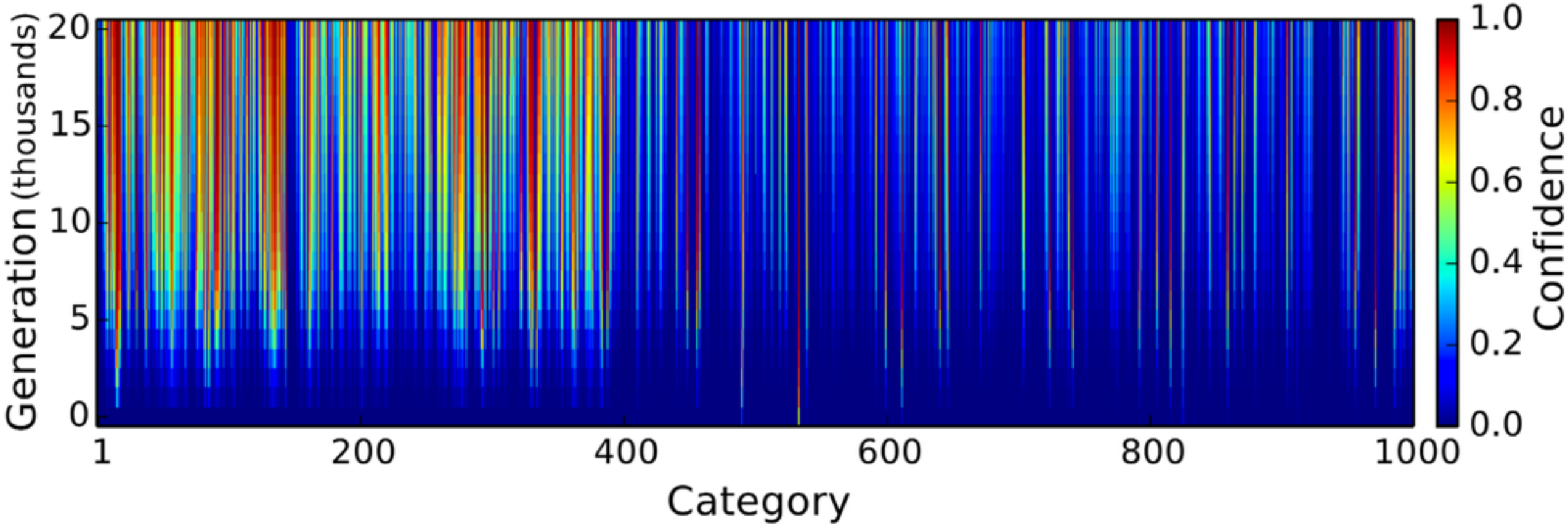
LeNet: 99.99% median confidence, 200 generations.

MNIST - Regular Images



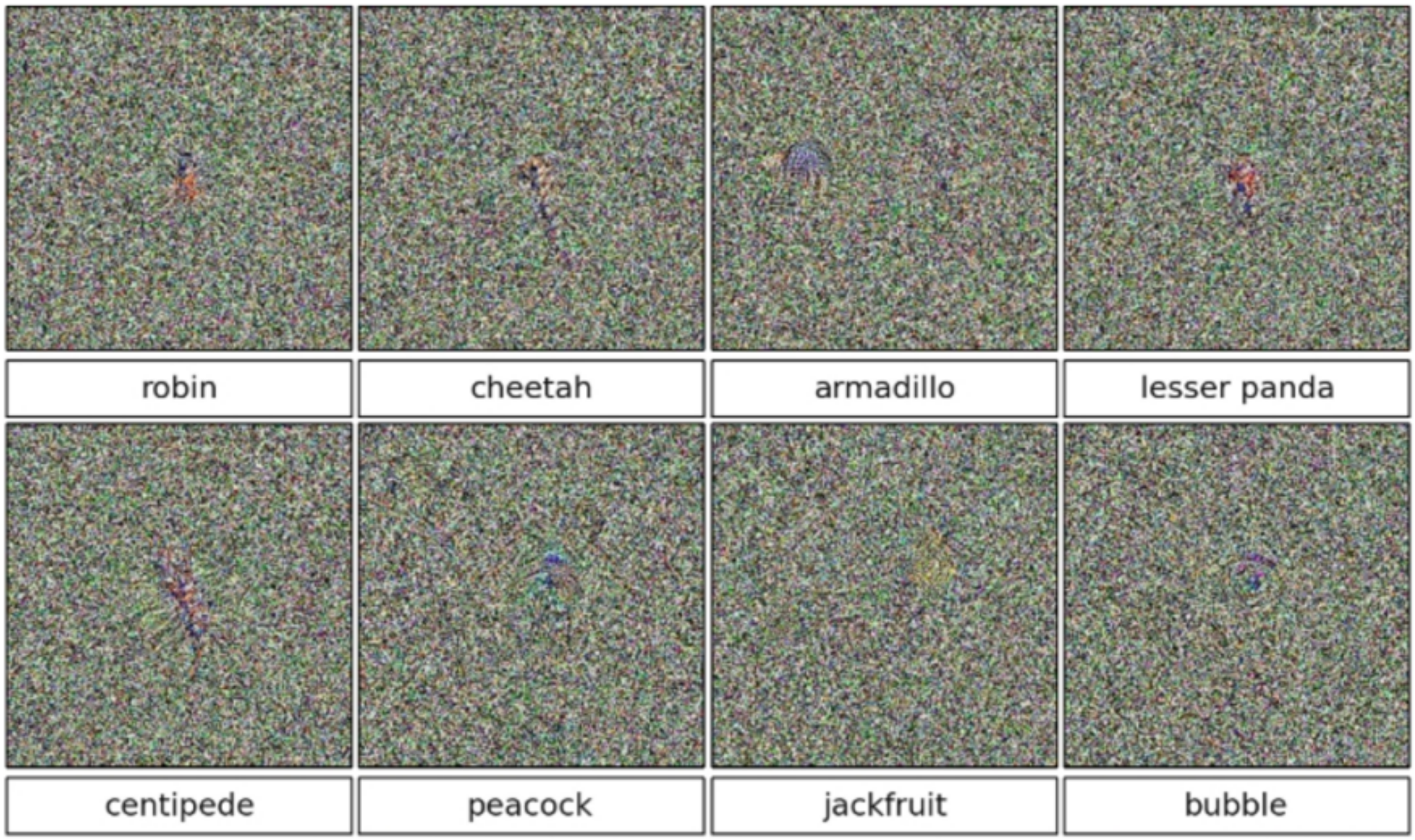
LeNet: 99.99% median confidence, 200 generations.

ImageNet - Irregular Images

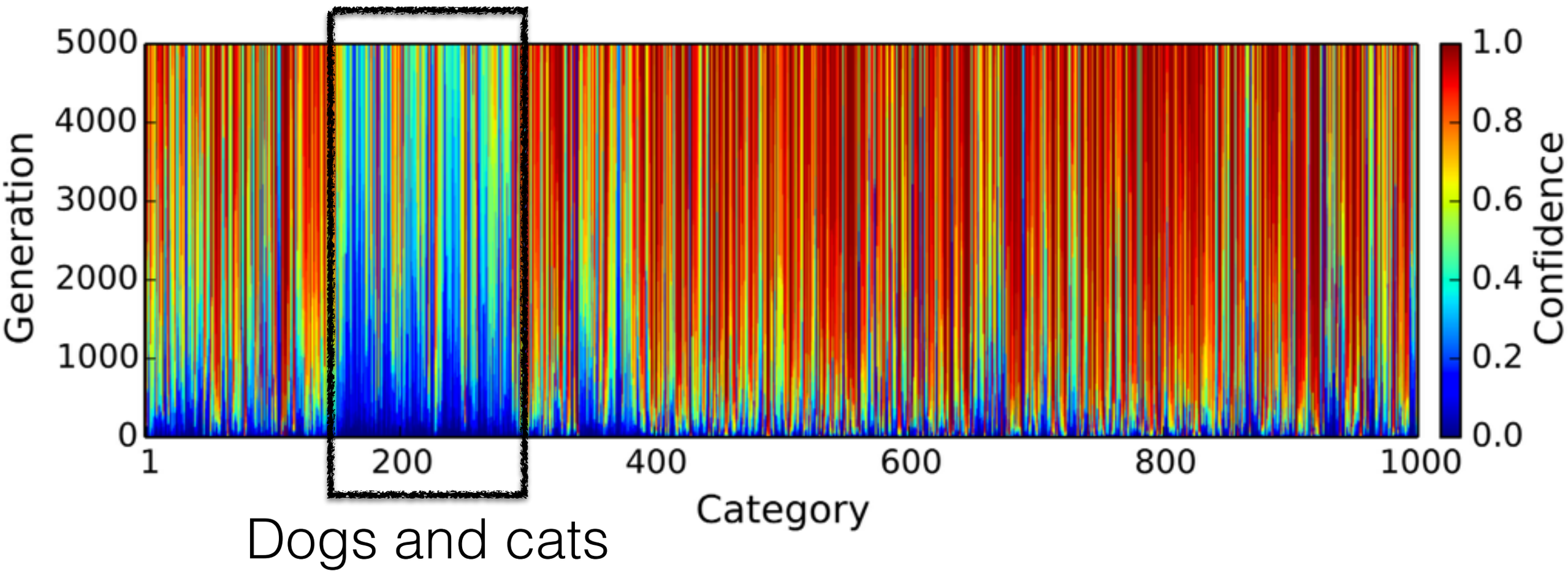


AlexNet: 21.59% median confidence, 20000 generations.
45 classes: > 99% confidence.

ImageNet - Irregular Images



ImageNet - Regular Images

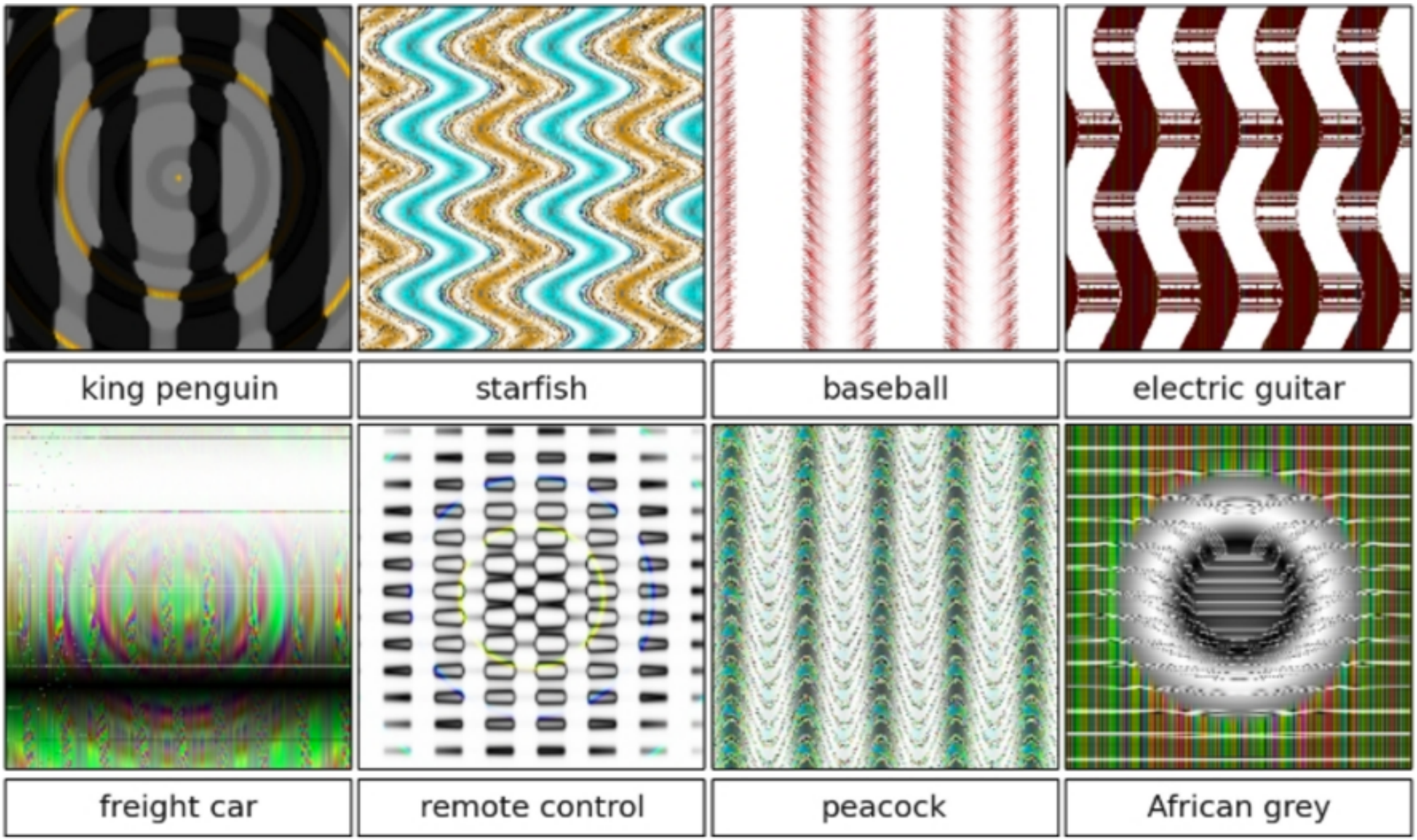


AlexNet: 88.11% median confidence, 5000 generations.
High confidence images are found in most classes.

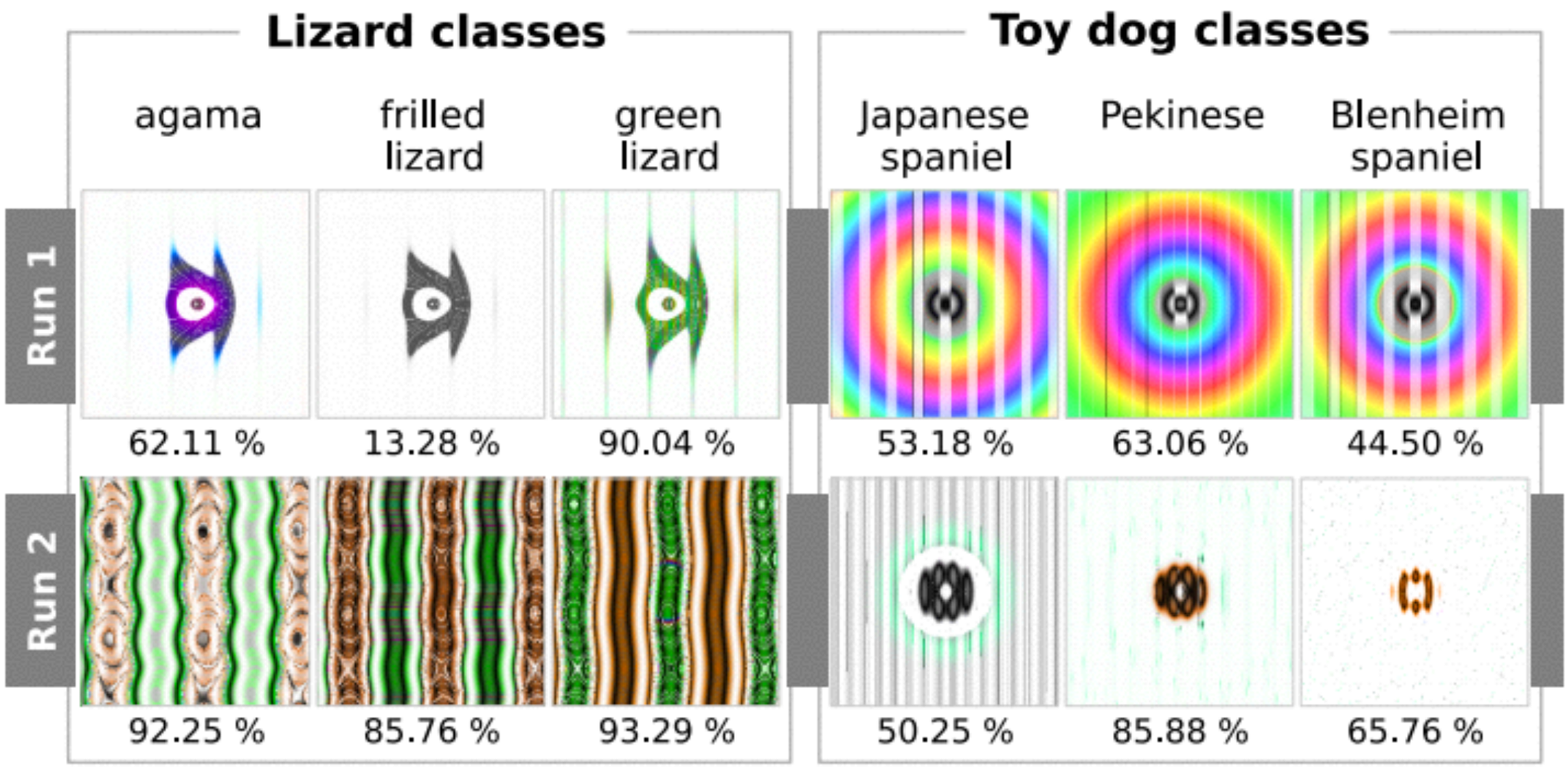
Difficulties in Dogs and Cats

- Size of dataset of cats and dogs is large.
  - Less overfit -> difficult to fool.
- Too many classes for cats and dogs.
  - e.g. difficult to achieve high score in Dog A while guaranteeing low score in Dog B.
  - [Recall] For the final softmax layer, it is difficult to give high confidence in the above case.

ImageNet - Regular Images



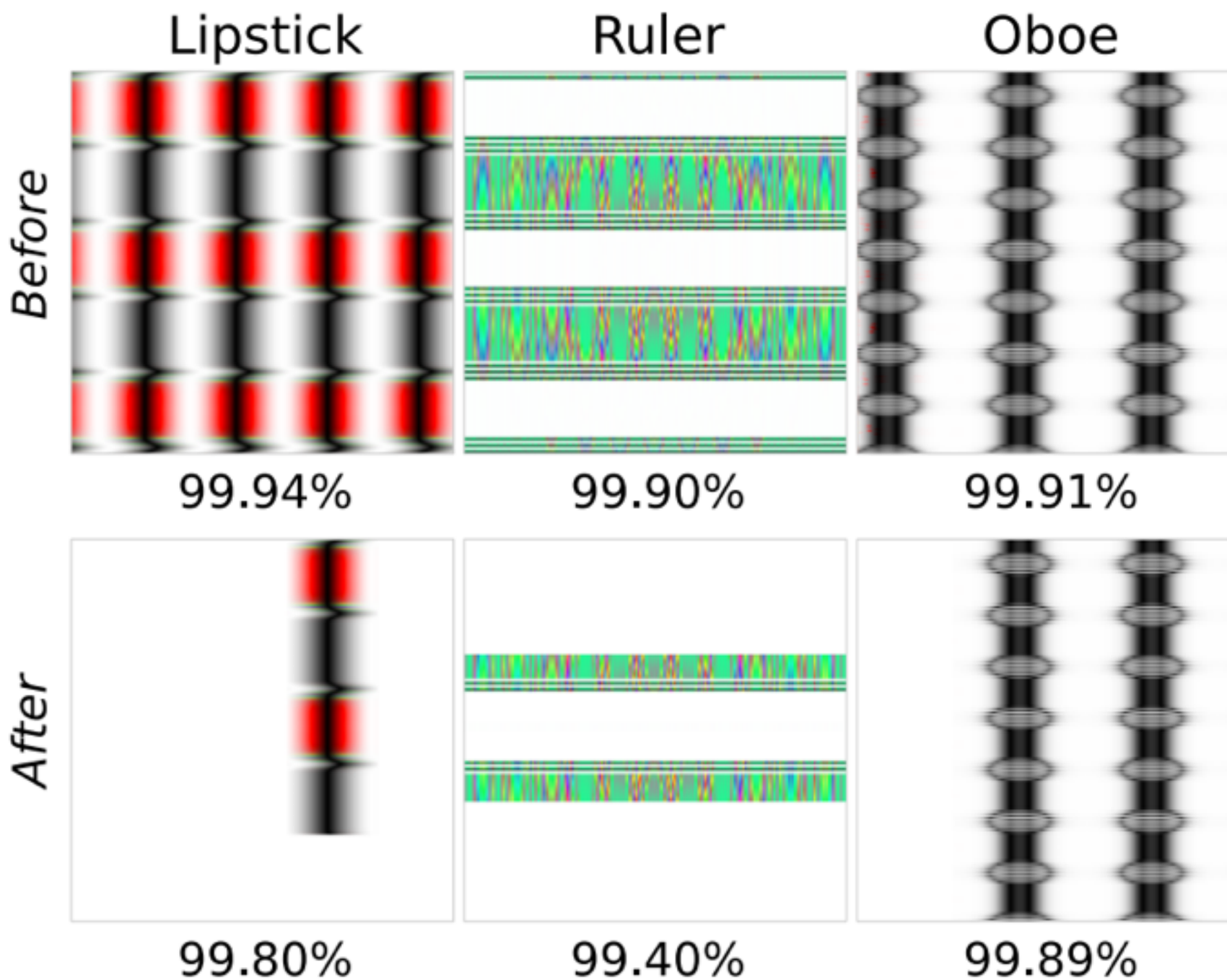
Fooling Closely Related Classes



Fooling Closely Related Classes

- Two possibilities:
  - [Recall] Imperceptible changes can change a DNN's class label. Evolution could produce very similar images to fool multiple classes.
  - Many of the images are related to each other naturally.
    - Different runs produce different images: many ways to fool the DNN.

Repetition of Patterns



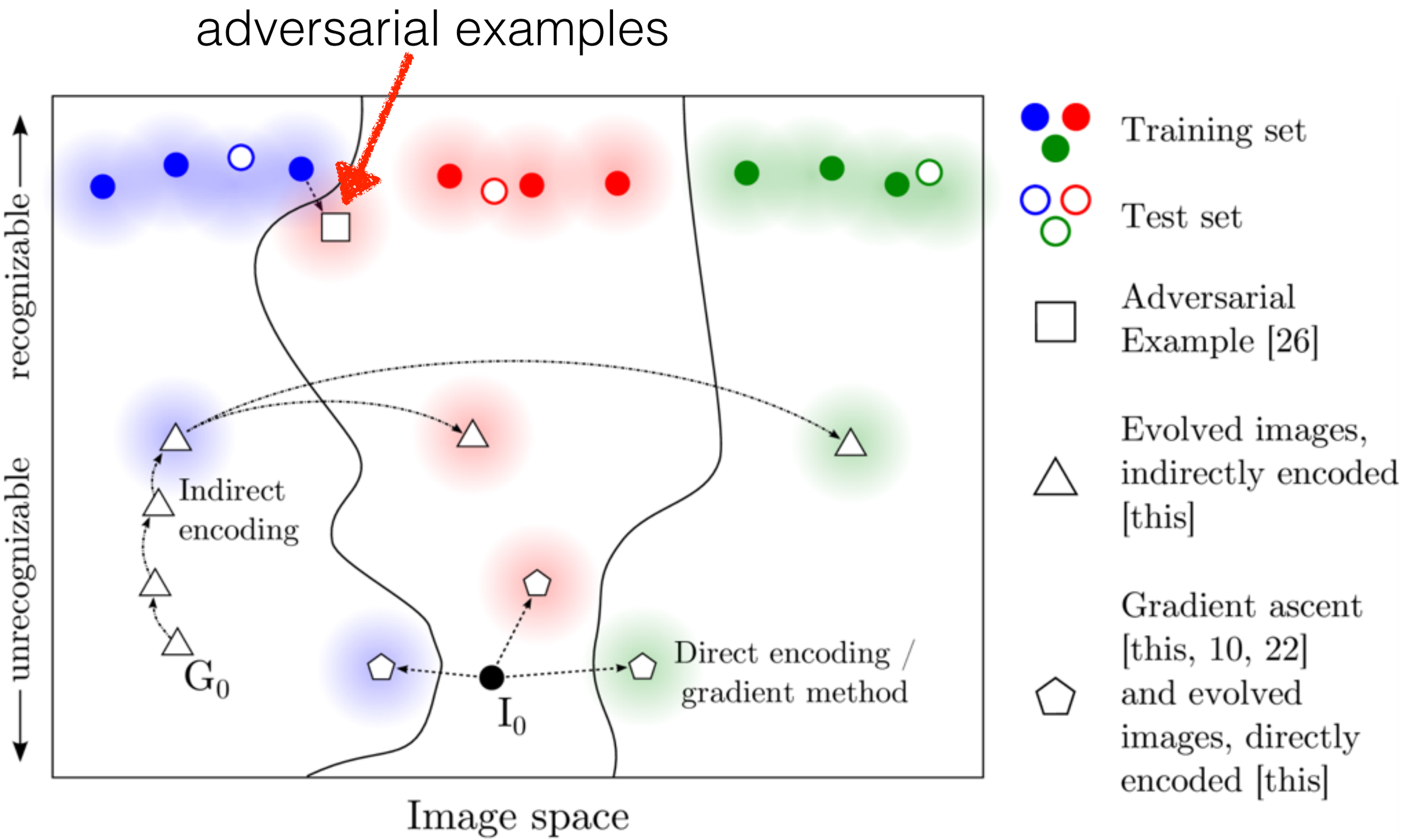
Repetition of Patterns

- Explanations
  - Extra copies make the DNN more confident.
  - DNNs tend to learn low&mid-level features rather than the global structure.
  - Many natural images do contain multiple copies.

Training with Fooling 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

Retraining does not help.


adversarial examples
recognizable
unrecognizable
Indirect encoding
G_0
I_0
Direct encoding / gradient method
Image space
Training set
Test set
Adversarial Example [26]
Evolved images, indirectly encoded [this]
Gradient ascent [this, 10, 22] and evolved images, directly encoded [this]

Why Do Adversarial Examples Exist?

- Past explanations
  - extreme nonlinearity of DNN.
  - insufficient model averaging.
  - insufficient regularization.
- New explanation
  - Linear behavior in high-dimensional spaces is sufficient to cause adversarial examples.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples." arXiv preprint arXiv:1412.6572 (2014).

Linear Explanations of Adversarial Examples

Perturbation: $\boldsymbol{\eta}$

Adversarial examples: $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}$

Pixel value precision: ϵ typically =1/255

Perturbation is meaningless if: $||\boldsymbol{\eta}||_\infty < \epsilon$

Activation of adversarial examples: $\boldsymbol{w}^\top \tilde{\boldsymbol{x}} = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{w}^\top \boldsymbol{\eta}$

$\boldsymbol{\eta} = \epsilon \mathrm{sign}(\boldsymbol{w})$ maximizes the increase of activation.

Linear Explanations of Adversarial Examples

Activation of adversarial examples: $\boldsymbol{w}^\top \tilde{\boldsymbol{x}} = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{w}^\top \boldsymbol{\eta}$

Assume the magnitude of the weight vector is **m** and the dimension is **n**:

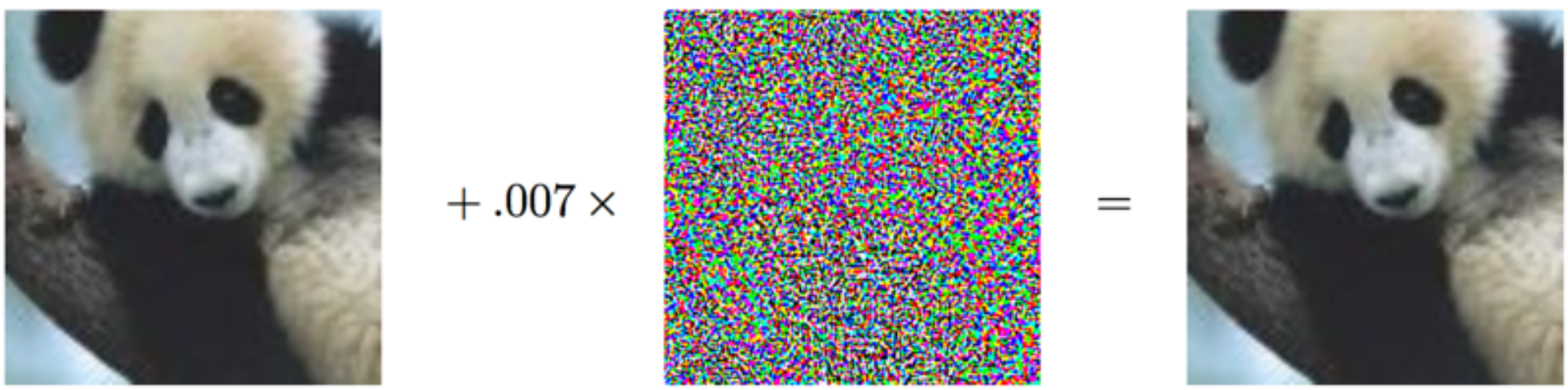
Increase of activation is: $\epsilon m n$

A simple linear model can have adversarial examples as long as its input has sufficient dimensionality.

Faster Way to Generate Adversarial Examples

Cost function: $J(\boldsymbol{\theta}, \boldsymbol{x}, y)$

Perturbation: $\boldsymbol{\eta} = \epsilon \operatorname{sign}\left(\nabla \boldsymbol{x} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right)$



$\boldsymbol{x}$

"panda"
57.7% confidence

$\operatorname{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$\boldsymbol{x} +$
$\epsilon \operatorname{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Faster Way to Generate Adversarial Examples

	epsilon	error rate	confidence
shallow softmax (MNIST)	0.25	99.9%	79.3%
maxout network	0.25	89.4%	97.6%
convolutional maxout network (CIFAR-10)	0.1	87.15%	96.6%

Adversarial Training of Linear Models

Simple case: Linear Regression.

$$P(y = 1 \mid \boldsymbol{x}) = \sigma\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right)$$

Train gradient descend on:

$$\mathbb{E}_{\boldsymbol{x}, y \sim p_{\text{data}}} \zeta(-y(\boldsymbol{w}^\top \boldsymbol{x} + b))$$

Adversarial training version is:

$$\mathbb{E}_{\boldsymbol{x}, y \sim p_{\text{data}}} \zeta(y(\epsilon ||\boldsymbol{w}||_1 - \boldsymbol{w}^\top \boldsymbol{x} - b))$$

Adversarial Training of Deep Networks

Regularized cost function:

$$\tilde{J}(\boldsymbol{\theta}, \boldsymbol{x}, y) = \alpha J(\boldsymbol{\theta}, \boldsymbol{x}, y) + (1 - \alpha) J(\boldsymbol{\theta}, \boldsymbol{x} + \epsilon \operatorname{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)))$$

On MNIST: error rate drops from 0.94% to 0.84%

For adversarial examples: error rate drops from 89.4% to 17.9%

Original model 40.9%

$\xrightarrow{\qquad\qquad}$

adversarial examples

$\xleftarrow{\qquad\qquad}$

Adversarially trained model 19.4%

Explaining Why Adversarial Examples Generalize

- [Recall] An adversarial example generated for one model is often misclassified by other models.

- When different models misclassify an adversarial examples, they often agree with each other.

- As long as $\boldsymbol{w}^\top \boldsymbol{\eta}$ is positive, adversarial examples work.

- Hypothesis: neural networks trained all resemble the linear classifier learned on the same training set.

  - Such stability of underlying classification weights causes the stability of adversarial examples.

Fooling Examples

- Can simply generate fooling examples by generating a point far from the data with larger norms (more confidence)

- Gaussian fooling examples:

  - softmax top layer: error rate: 98.35%, average confidence: 92.8%.

  - independent sigmoid top layer: error rate: 68%, average confidence: 87.9%.

Summary

- **Intriguing properties**
  - No difference between individual high level units and random linear combinations of high level units.
  - Adversarial Examples
    - Indistinguishable.
    - Generalize.
- **Fooling images**
  - Generate fooling images via evolution.
    - Direct encoding and indirect encoding (irregular and regular images).
  - Retraining does not boost immunity.

Generative Adversarial Nets

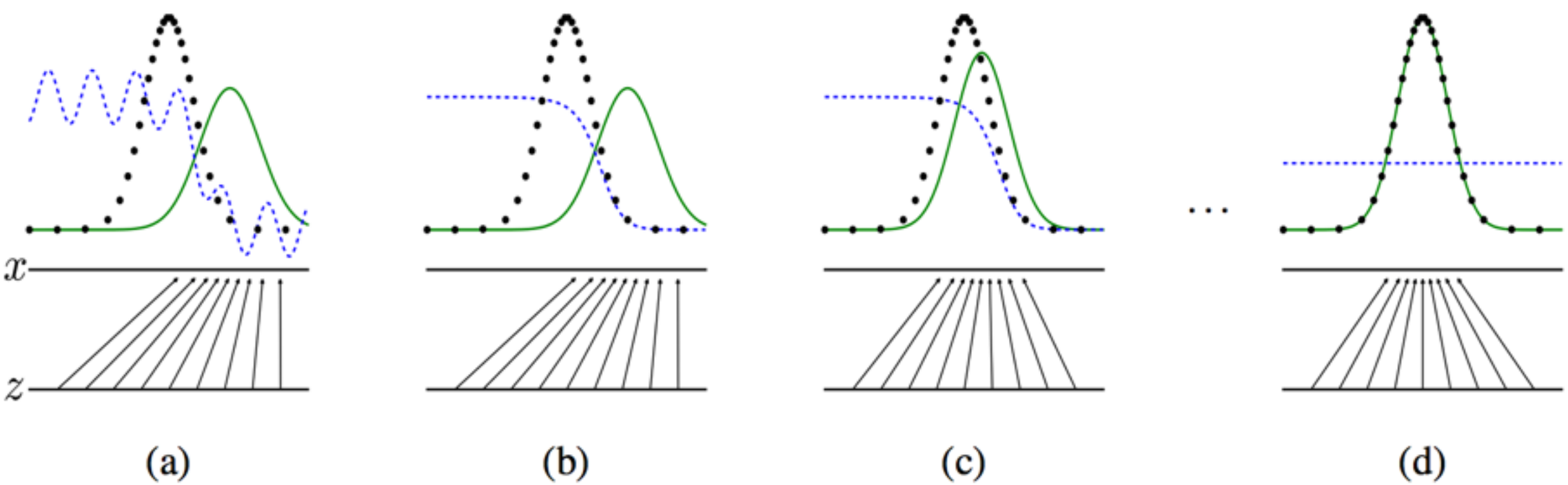
- Two types of models:
  - Generative model: generative model learns the joint probability distribution of the data - $p(x, y)$.
  - Discriminative model: discriminative model learns the conditional probability distribution of the data - $p(y \mid x)$.
  - Much easier to get discriminative model with the generative model.

Main Idea

- Adversarial process:
  - simultaneously train two models
    - a generative model G captures the data distribution.
    - discriminative model D - tells whether a sample comes from the training data or not.
  - Optimal solution:
    - G recovers the data distribution.
    - D is 1/2 everywhere.

Two-player minmax game

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$



Thanks.